BHARATI VIDYAPEETH'S

INSTITUTE OF COMPUTER APPLICATIONS & MANAGEMENT

(Affiliated to Guru Gobind Singh Indraprastha University, Approved by AICTE, New Delhi)

Operating Systems with Linux Lab (MCA-163) Practical File

Submitted To:

Dr. Sunil Pratap Singh

(Associate Professor)

Submitted By:

Manpreet Singh (030)

MCA 1st Sem, Sec 1

INDEX

S. No.	Problem Description	Date of Execution	Sign.
P1	Install VirtaulBox and then configure Linux (Ubantu) in VirtualBox.	15/11/2023	
P2	Run ps and note the PID of your shell. Log out and log in again and run ps again. What do you observe?	15/11/2023	
P3	Enter the following commands, and note your observations: (i) who and tty, (ii) tput clear, (iii) id, (iv) ps and echo.	15/11/2023	
P4	Run the following commands, and then invoke ls. What do you conclude? echo > README [Enter] echo > readme [Enter]	15/11/2023	
P5	Create a directory, and change to that directory. Next, create another directory in the new directory, and then change to that directory too. Now, run $ cd without any arguments followed by pwd. What do you conclude?	15/11/2023	
P6	Create a file mca containing the words “Hello MCA Class!”. Now create a directory bvicam, and then run mv mca bvicam. What do you observe when you run both ls and ls	15/11/2023	

	bar?		
P7	Run $ who am i and then interpret the output.	15/11/2023	
P8	Find out whether the following commands are internal or external: echo, date, pwd, and ls.	15/11/2023	
P9	Display the current date in the form dd/mm/yyyy.	15/11/2023	

P10	Both of the following commands try to open the file mca, but the error messages are a little different. What could be the reason? (CO2)
	$ cat mca
	cat: mca: No such file or directory
	$ cat < mca
	bash: mca: No such file or directory
		15/11/2023	

P11	Run the following commands, and discuss their output? (CO2) 1. $ uname 2. $ passwd 3. $ echo $SHELL	15/11/2023	

	1. $ man man 2. $ which echo 3. $ type echo 4. $ whereis ls 5. $ cd 6. $ cd $HOME 7. $ cd ~		
P12	Frame ls command to (i) mark directories and executables separately, and (ii) also display hidden files.	15/11/2023	

P13	Find out the result of following: $ cat mca mca mca	15/11/2023	
P14	Run the following and determine which commands will work? Explain with reasons. (CO2) 1. $ mkdir a/b/ 2. $ mkdir a a/b 3. $ rmdir a/b/c	15/11/2023	
	1. $ rmdir a a/b 2. $ mkdir /bin/mca		
P15	How does the command mv mca1 mca2 behave, where both mca1 and mca2 are directories, when (i) mca2 exists, (ii) mca2 doesn‟t exist?	15/11/2023	

P16	Assumingthatyouarepositionedinthedirectory /home/bvicam, what are these commands presumed to do, and explain whether they will work at all: (a) $ cd ../.. 1. $ mkdir ../bin 2. $ rmdir .. 3. $ ls ..	17/11/2023	
P17	Apply Peterson algorithm for solving the critical section problem with C/Java multi-threaded programming. Assume appropriate code snippet for critical section.	17/11/2023	
P18	Apply Bakery algorithm for synchronization of processes/threads in a C/Java program. Assume appropriate code snippet for critical section.	17/11/2023	
P19	Write C/Java program to simulate and solve the Producer-	17/11/2023	

	Consumer problem.		

P20	Implement Semaphore(s) in aC/Java-multithreaded program to simulate the working and solution of Reader-Writer problem. Assume multiple readers and writers.	17/11/2023	
P21	Create a zombie process and an orphan process in a „C" program with appropriate system calls.	25/11/2023	
P22	Write a „C" program which creates a new process and allows both, child and parent, to report their identification numbers (ids). The parent process should wait for the termination of the child process.	25/11/2023	
P23	Write two „C" programs (A.c and B.c) where one program (A.c) creates a child process and then that child process executes the code of other program (B.c). The logic of program „B.c" is to generate all the prim number within the specified limit.	25/11/2023	
P24	Write an appropriate „C" program which implements the concept of dynamic memory allocation (use of malloc(), calloc(), realloc(), and free() system call).	25/11/2023	
P25	Create a text file, named as „courses.txt" that contains the	25/11/2023	

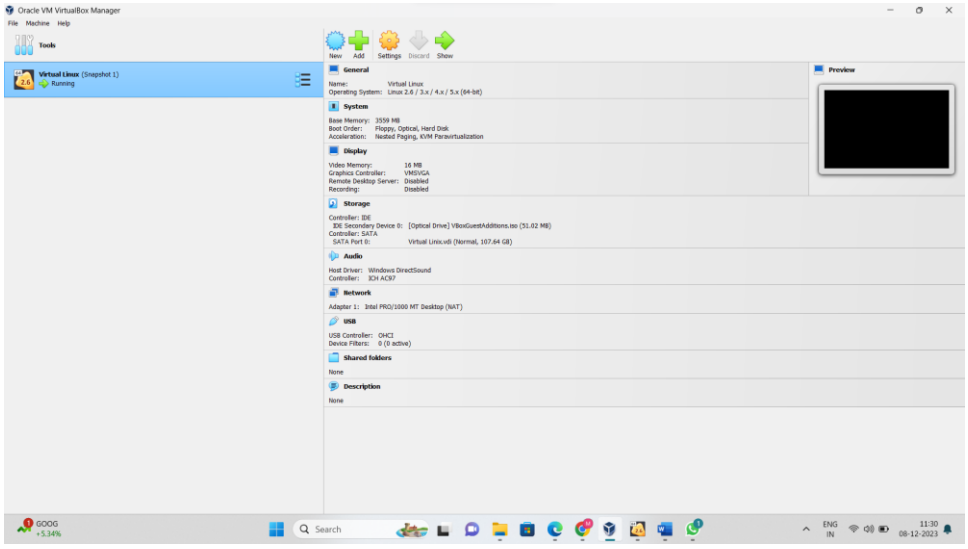
	following four lines: Java Programming Operating System Discrete Structure Write a „C‟ program that forks three other processes. After forking, the parent process goes into wait state and waits for the children to finish execution. Each child process reads a line from the „course.txt‟ file (Child 1 Reads Line 1, Child 2 Reads Line 2, and Child 3 Reads Line 3) and each prints the respective line. The lines can be printed in any order.		
P26	Write a „C‟ program (using appropriate system calls of Linux) that generates „n‟ integers and stores them in a text file, named as „All.txt‟. Then, retrieve the stored integers from this file and copy to “Odd.txt” and„ Even.txt‟ based upon the type of number, i.e. if the retrieved integer if odd number then store in „Odd.txt‟ file or if the retrieved integer is even then store in „Even.txt‟ file. Finally, display the contents of all three files on the screen.	01/12/2023	
P27	Write a program in „C‟ which accepts the file or directory name and permission (access rights) from the user and then changes the access rights accordingly. Use appropriate system call(s) of Linux.	01/12/2023	

P28	Write a „C" program (using appropriate system calls of Linux) which generates and stores the characters from „a" to „z". Then, display the stored characters in alternative manner, like: a, c, e, g, ..., etc.	01/12/2023	

P29	Write a „C" program (using appropriate system calls of Linux) which receives roll number and names of „n" students, from the user one-by-one and then stores them in a text file, named as „Student.txt". After inserting all „n" roll numbers and names, display the contents of file. Also, display the access rights of the file „Student.txt".	01/12/2023	
P30	Demonstrate the use of following system calls by writing an appropriate „C" program. (CO4) 1. lseek() 2. chmod() 3. umask() 4. access() 5. utime()	01/12/2023	

P1 Install VirtaulBox and then configure Linux (Ubantu) in VirtualBox.

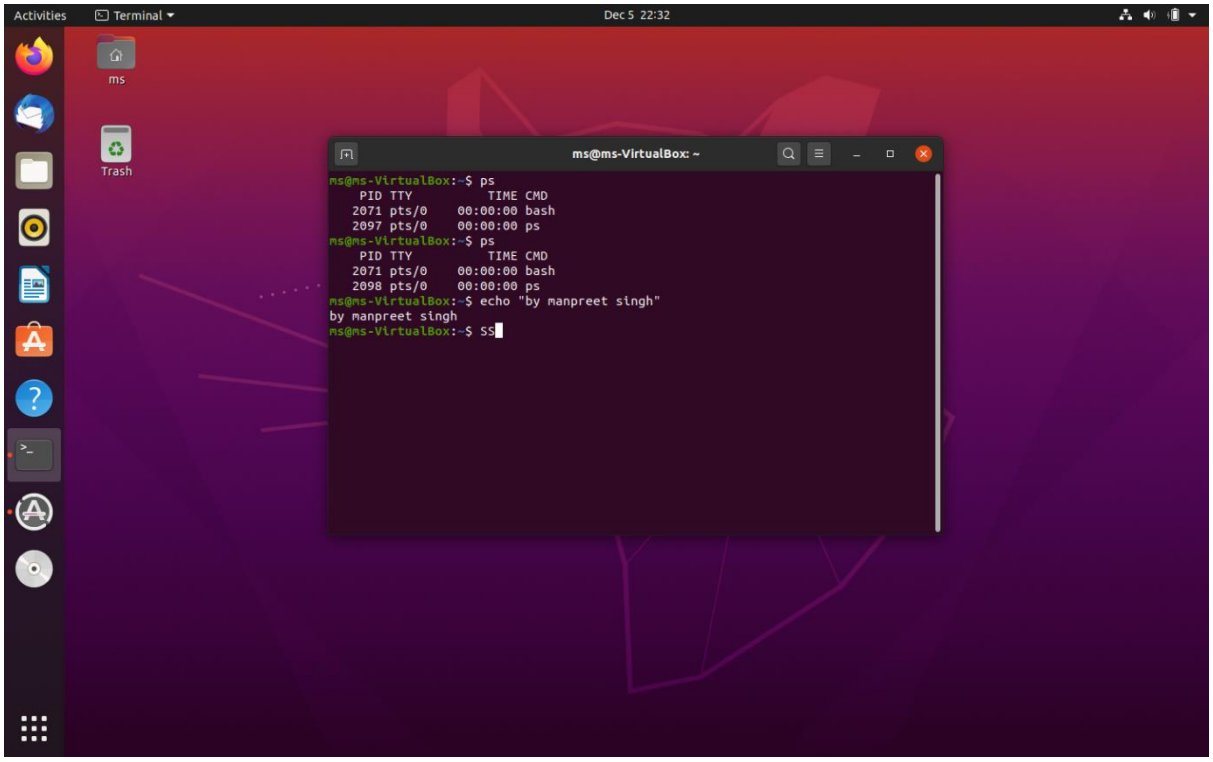
ANS



P2 Run ps and note the PID of your shell. Log out and log in again and run ps again. What do you observe?

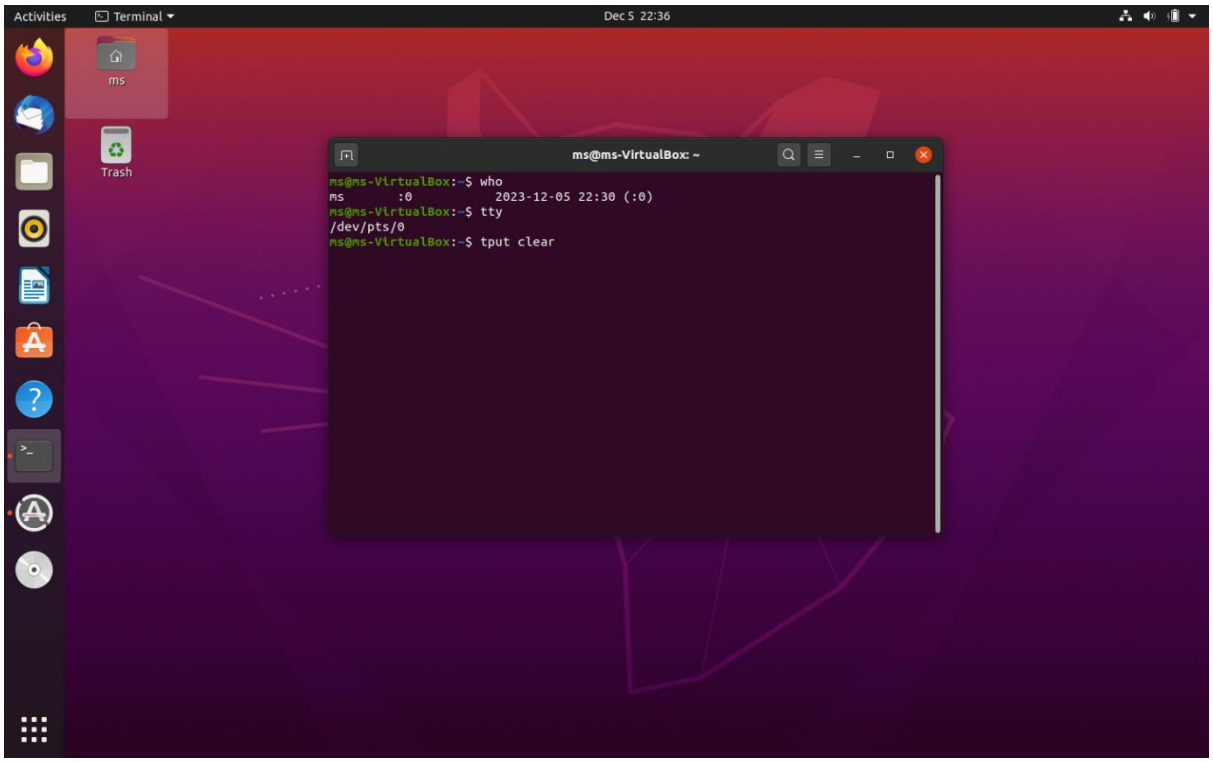
ANS

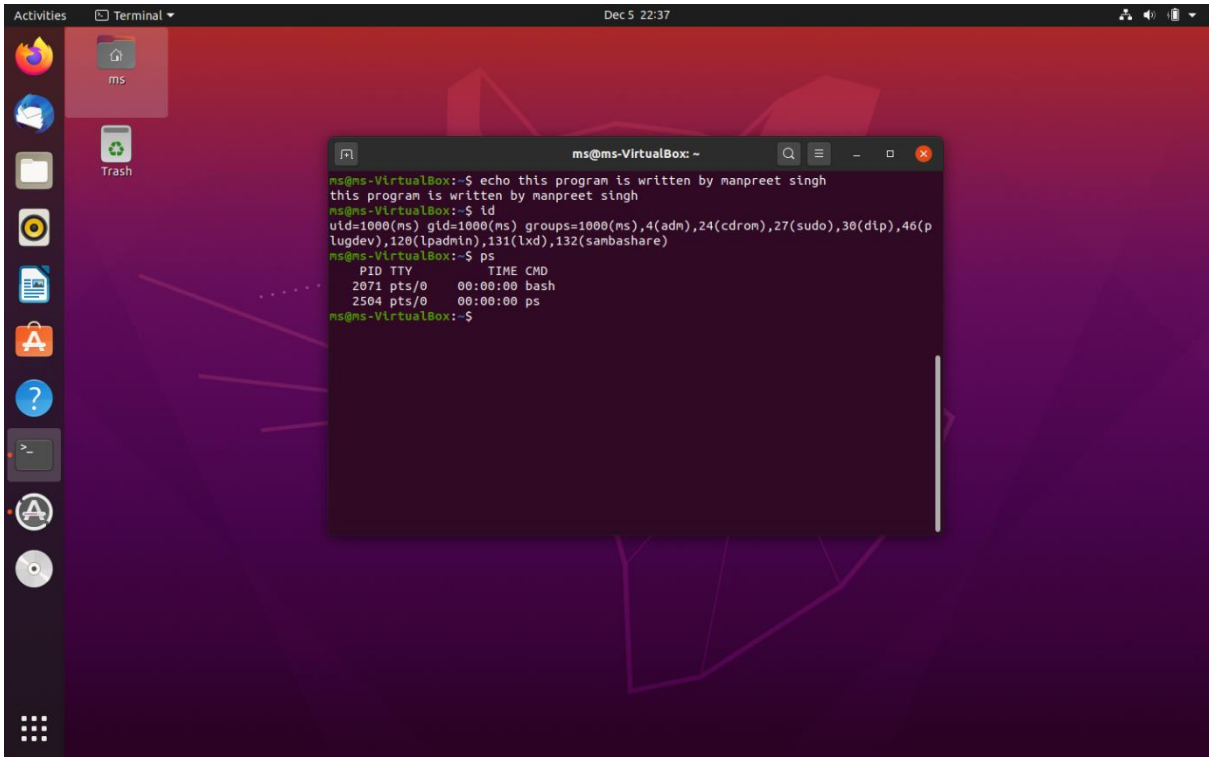
ps command generates important information regarding the tty Such as their pid(s), tty ,time and type of shell Each time we use ps it allocated our bash shell to a new memory location Hence the pids vary when we open and close our bash shell and reopen it.



P3. Enter the following commands, and note your observations: (i) who and tty, (ii) tput clear, (iii) id, (iv) ps and echo.

SCREENSHOT:



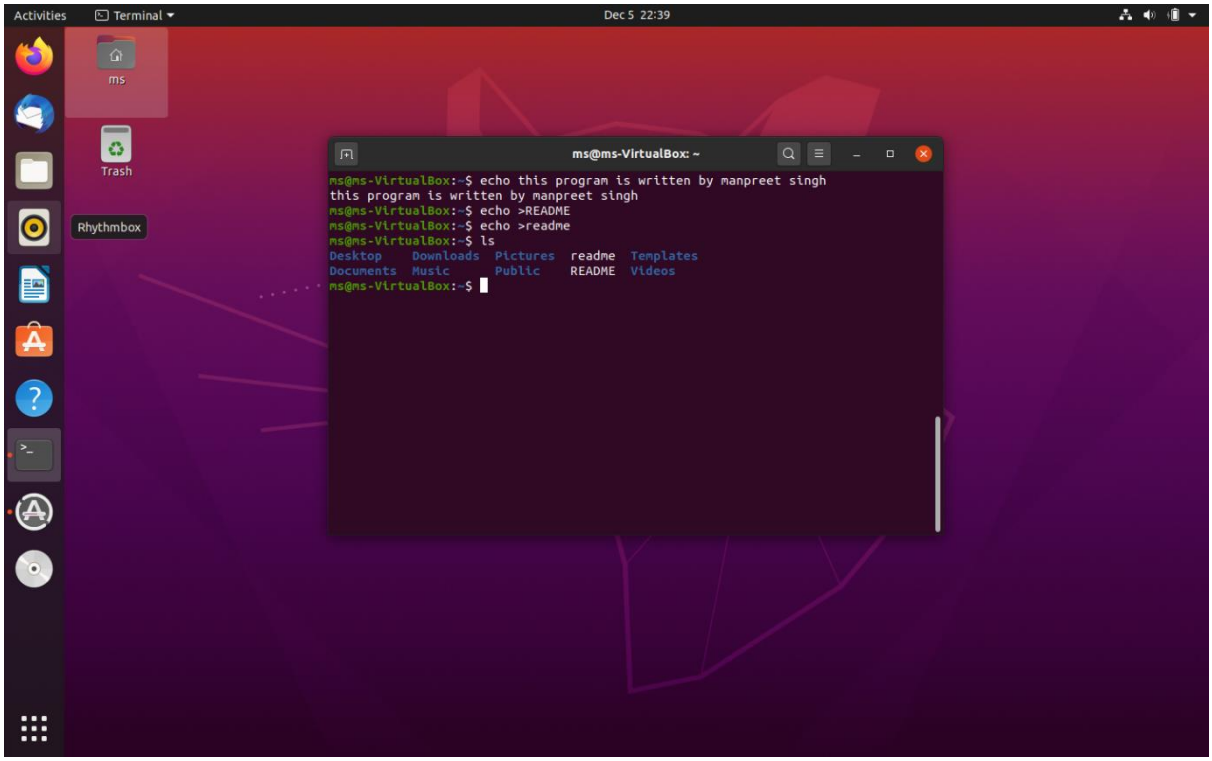


P4. Run the following commands, and then invoke ls. What do you conclude?

echo > README [Enter]

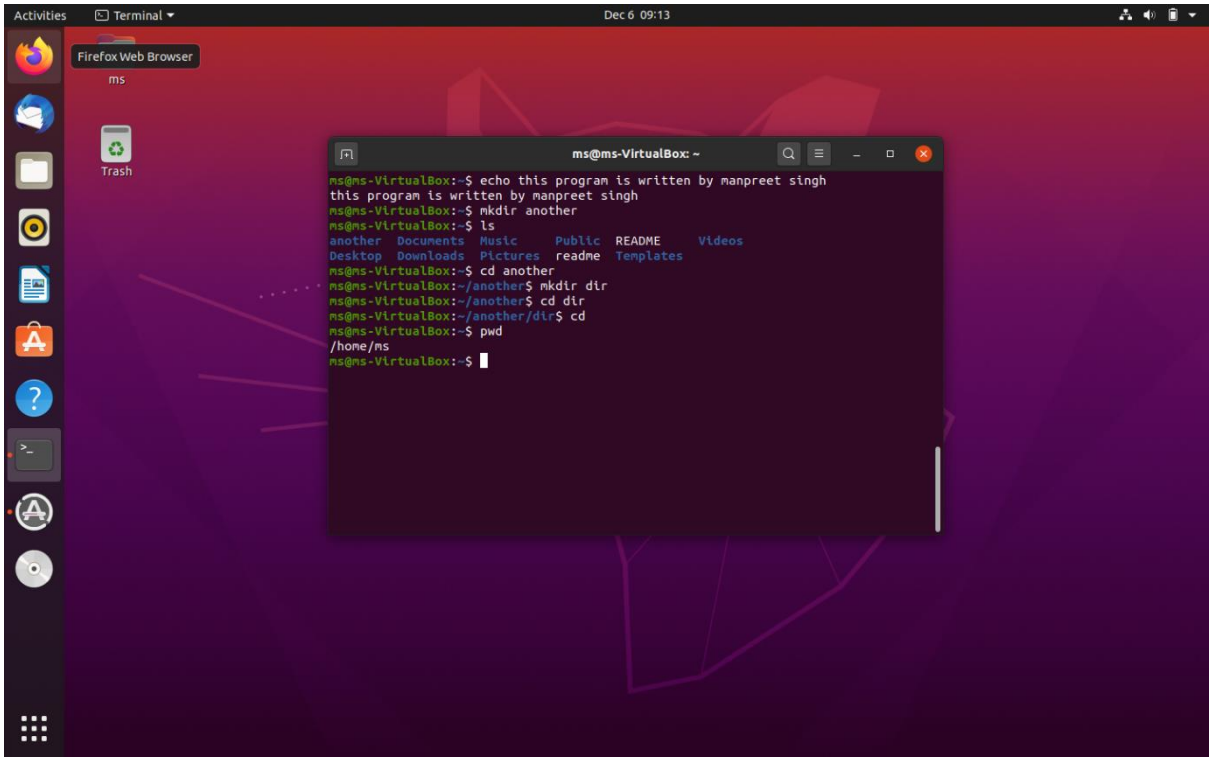
echo > readme [Enter]

SCREENSHOT:



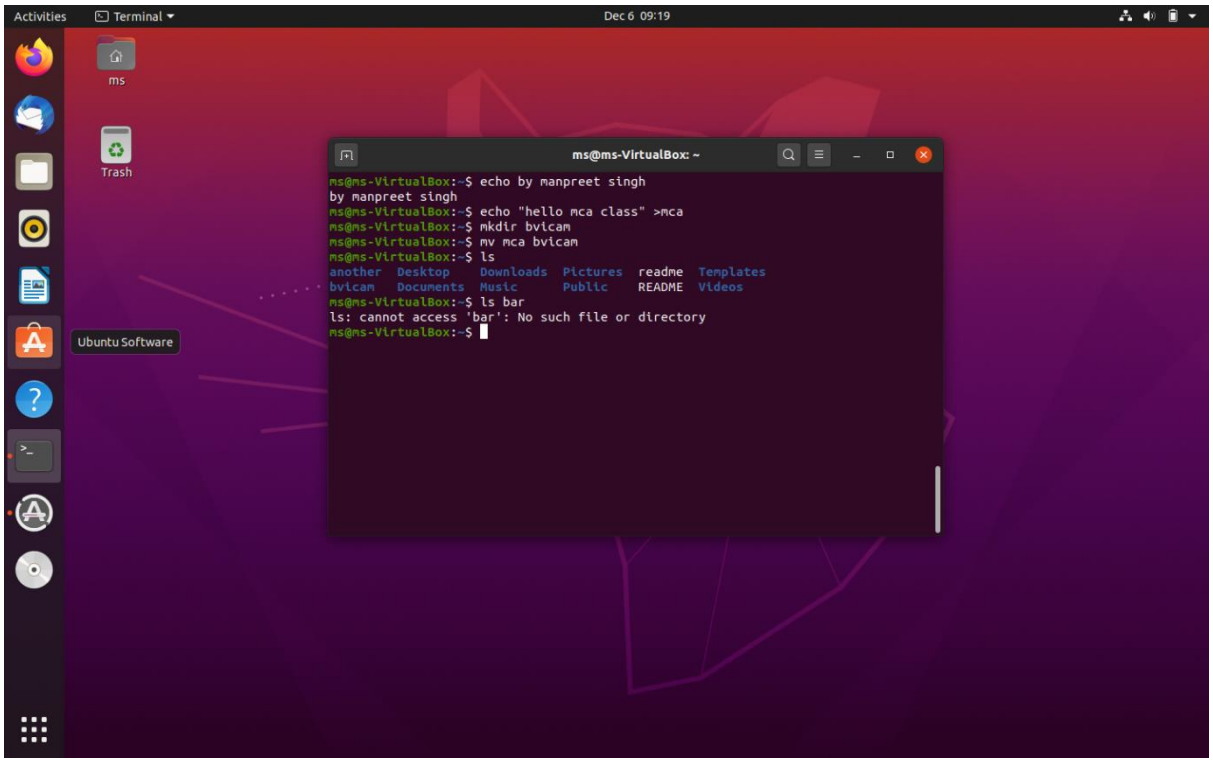
P5. Create a directory, and change to that directory. Next, create another directory in the new directory, and then change to that directory too. Now, run $ cd without any arguments followed by pwd. What do you conclude?

SCREENSHOT



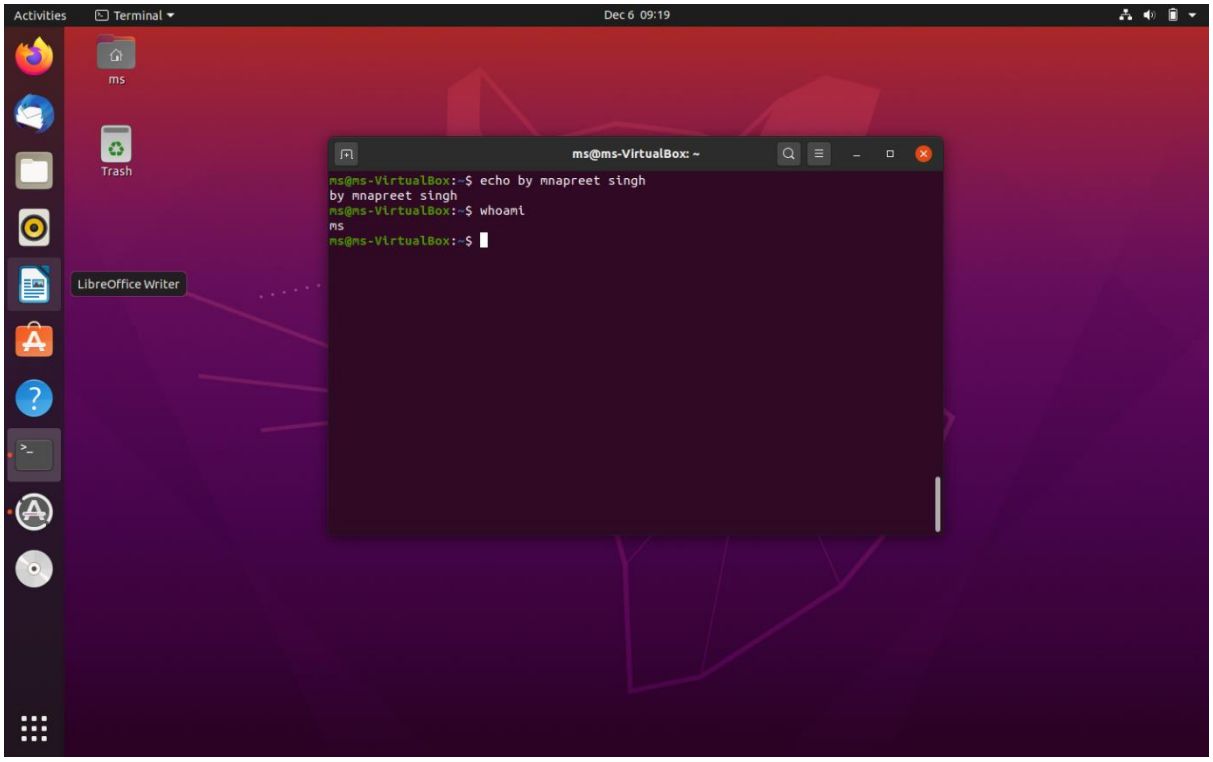
P6 Create a file mca containing the words “Hello MCA Class!”. Now create a directory bvicam, and then run mv mca bvicam. What do you observe when you run both ls and ls bar?

SCREENSHOT:



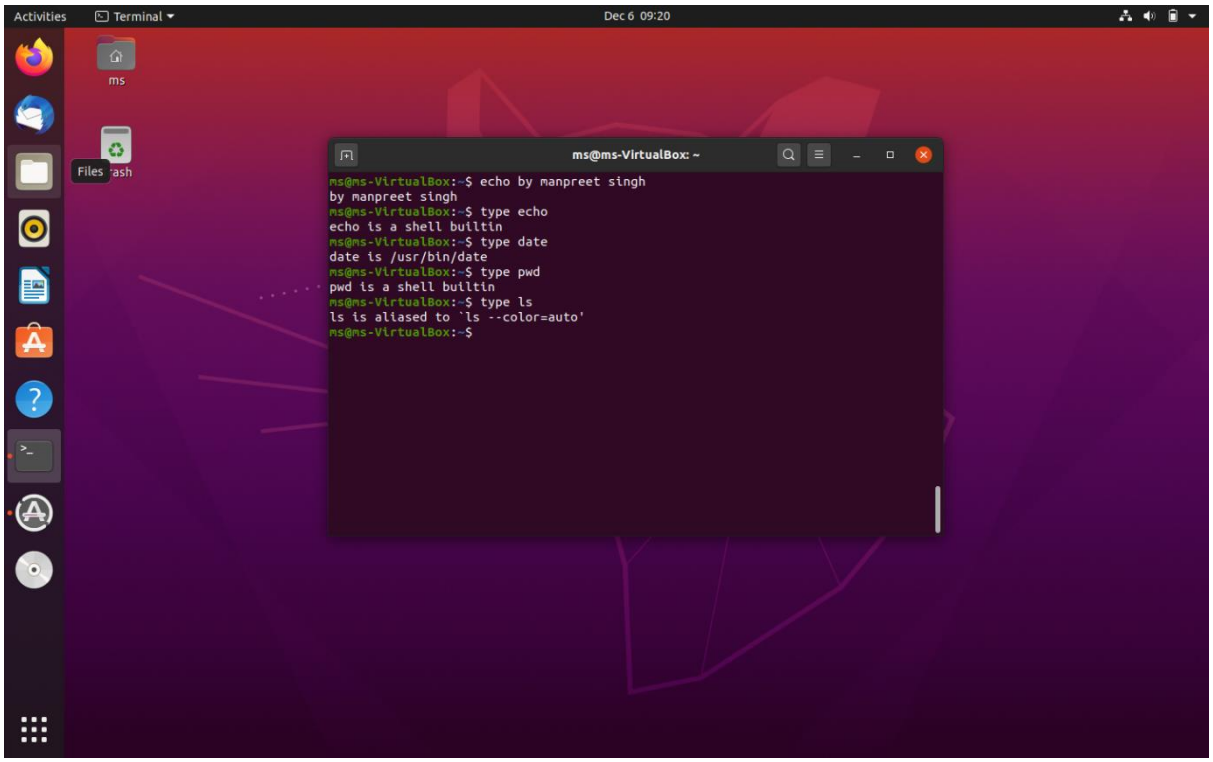
P7. Run $ who am i and then interpret the output.

SCREENSHOT:



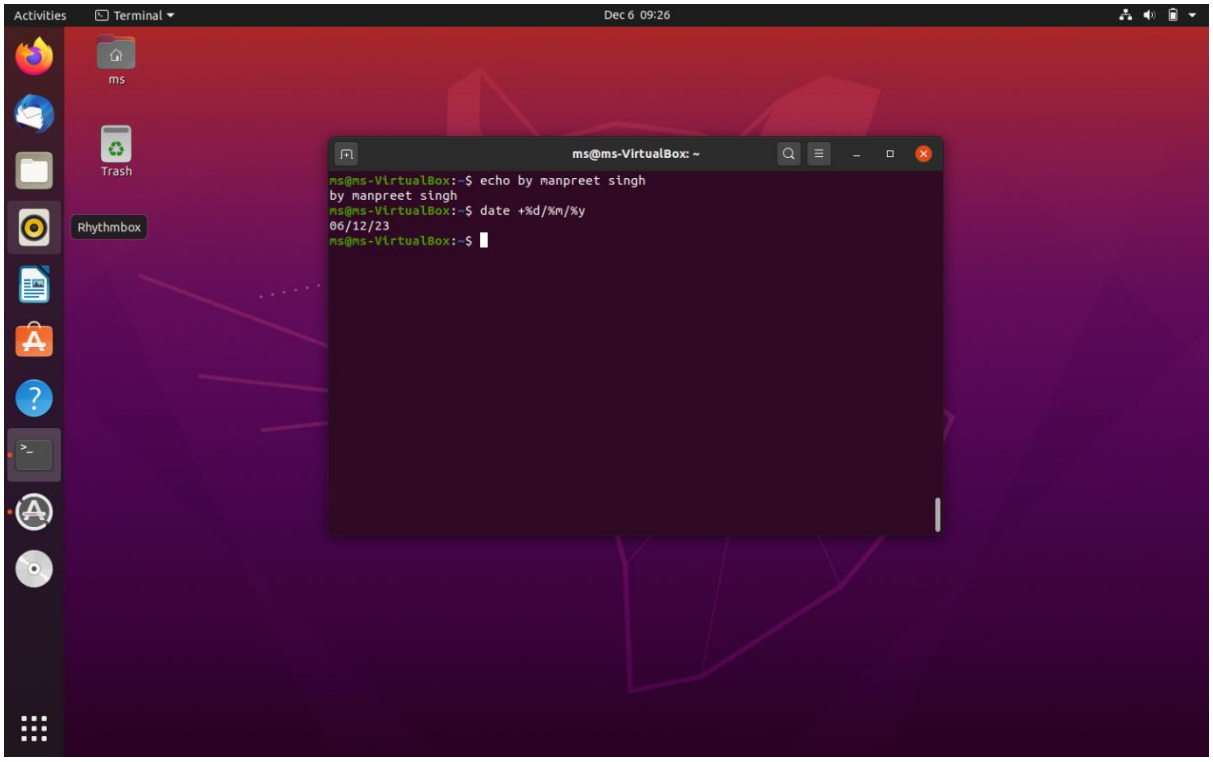
P8. Find out whether the following commands are internal or external: echo, date, pwd, and ls.

SCREENSHOT:



P9. Display the current date in the form dd/mm/yyyy.

SCREENSHOT:



P10. Both of the following commands try to open the file mca, but the error messages are a little different. What could be the reason? (CO2)

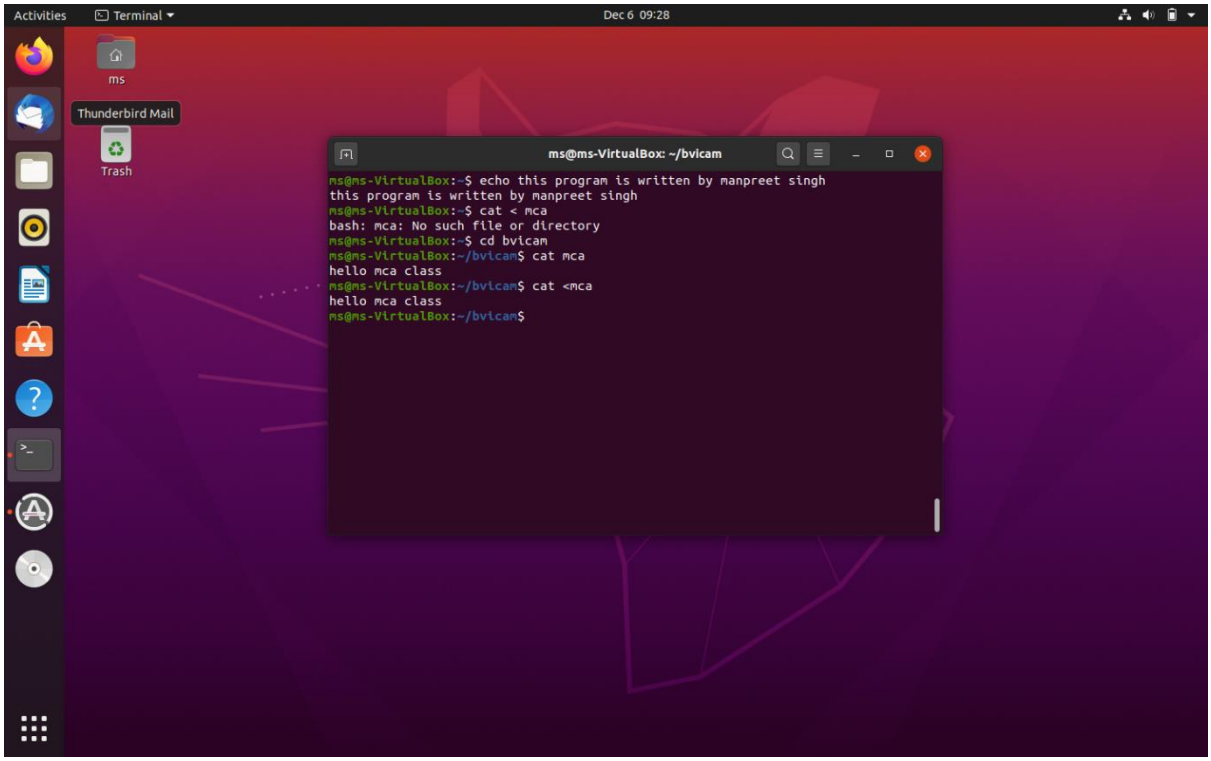
$ cat mca

cat: mca: No such file or directory

$ cat < mca

bash: mca: No such file or directory

SCREENSHOT:



P11. Run the following commands, and discuss their output?

(a) $ uname

(b) $ passwd

(c) $ echo $SHELL

(d) $ man man

(e) $ which echo

(f) $ type echo

(g) $ whereis ls

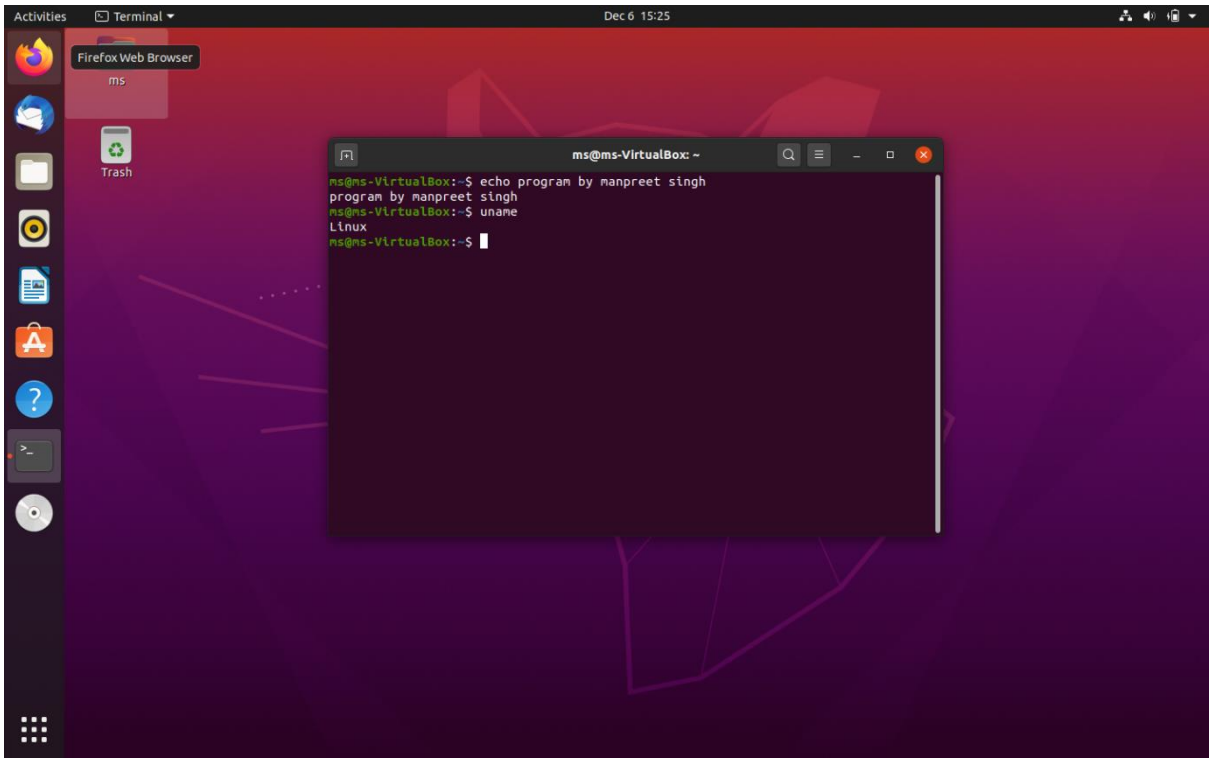
(h) $ cd

(i) $ cd $HOME

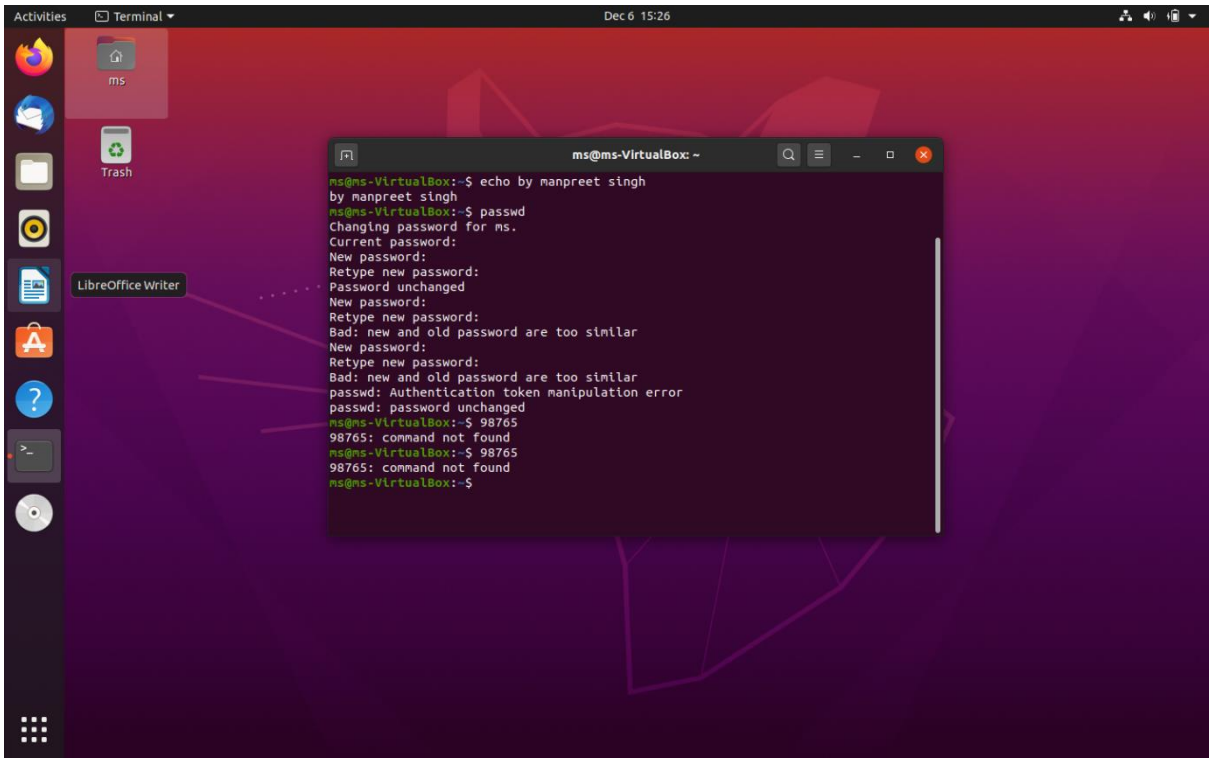
(j) $ cd ~SOLUTION:

SCREENSHOT:

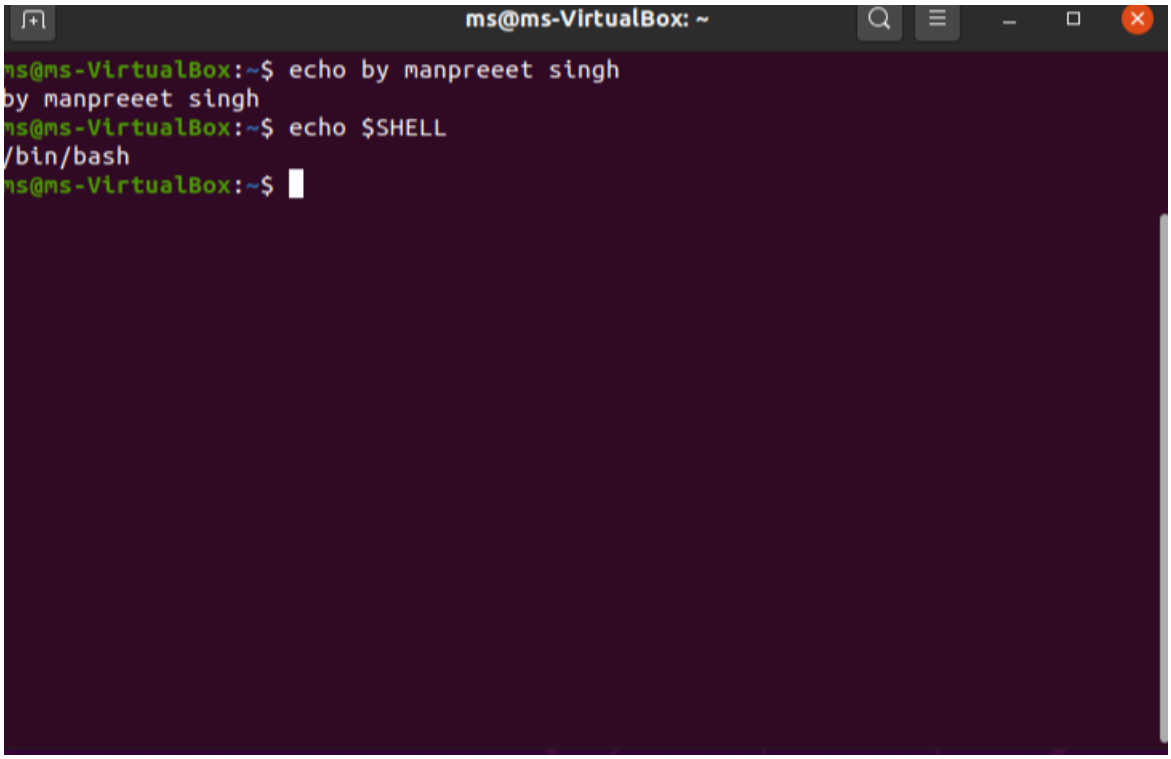
A



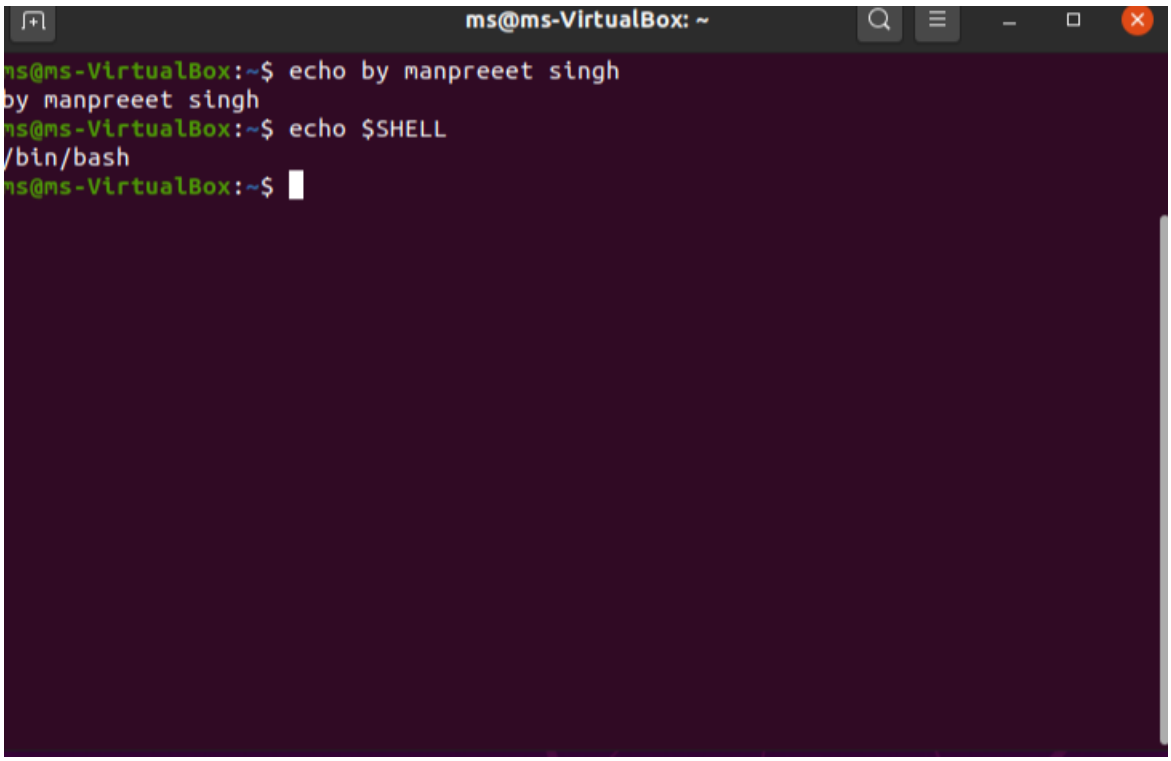
B



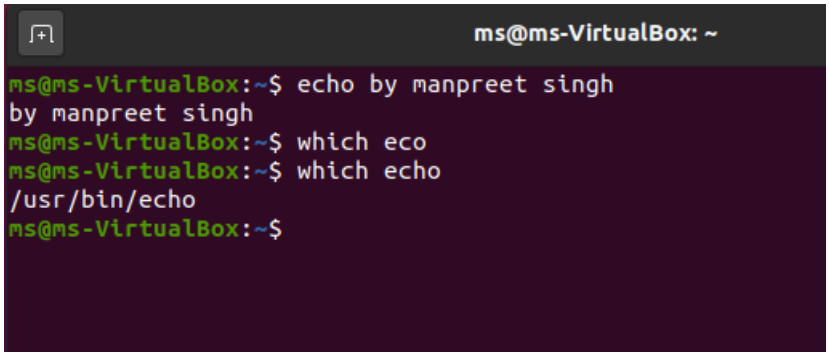
C


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ echo by manpreeet singh
by manpreeet singh
ms@ms-VirtualBox:~$ echo $SHELL
/bin/bash
ms@ms-VirtualBox:~$

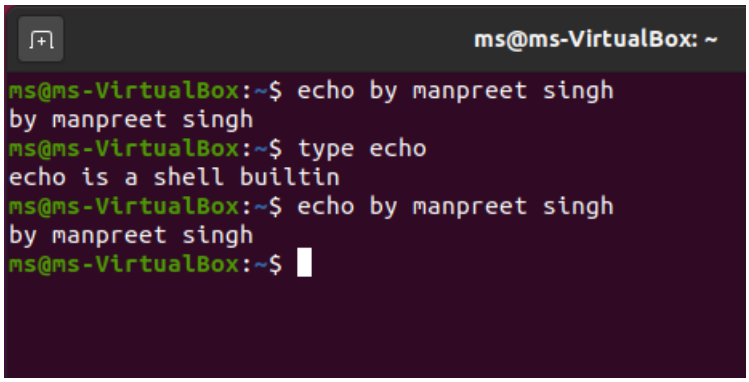
D


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ echo by manpreeet singh
by manpreeet singh
ms@ms-VirtualBox:~$ echo $SHELL
/bin/bash
ms@ms-VirtualBox:~$

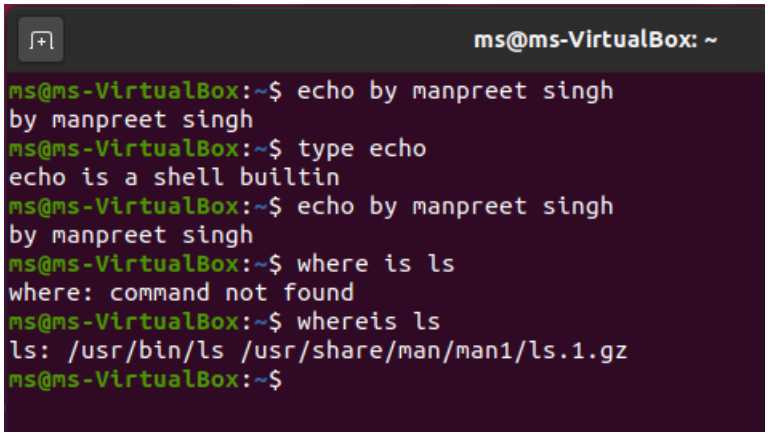
E


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ which eco
ms@ms-VirtualBox:~$ which echo
/usr/bin/echo
ms@ms-VirtualBox:~$

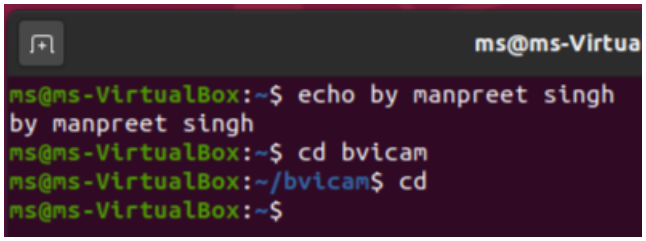
F


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ type echo
echo is a shell builtin
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$

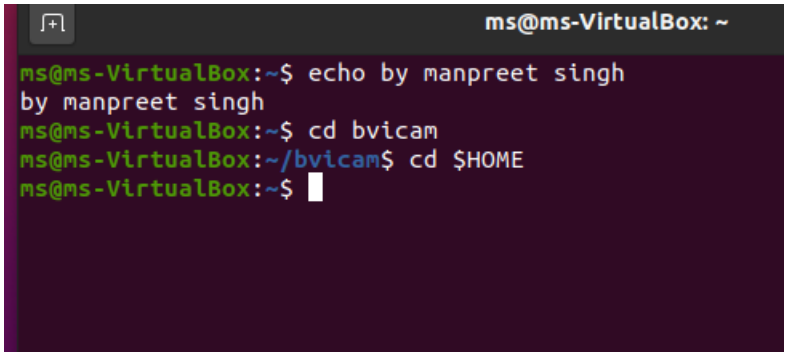
G


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ type echo
echo is a shell builtin
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ where is ls
where: command not found
ms@ms-VirtualBox:~$ whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
ms@ms-VirtualBox:~$

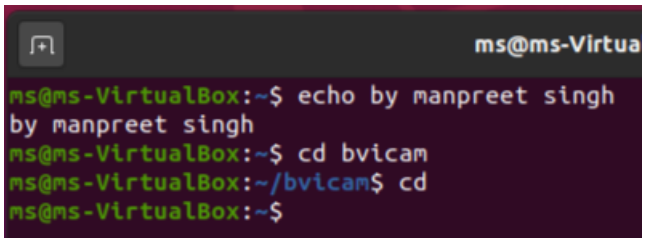
H


ms@ms-Virtua
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ cd bvicam
ms@ms-VirtualBox:~/bvicam$ cd
ms@ms-VirtualBox:~$

J



J



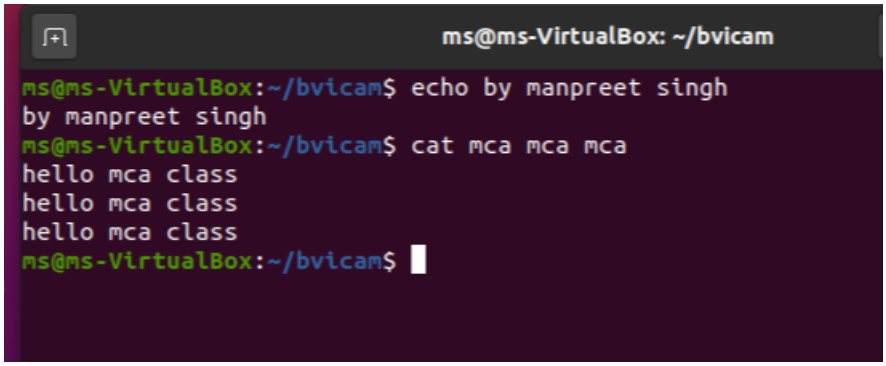
P12. Frame ls command to (i) mark directories and executables separately, and (ii) also display hidden files.

SCREENSHOT:

P13. Find out the result of following:

$ cat mca mca mca

SCREENSHOT:



P14. Run the following and determine which commands will work? Explain with reasons.

(a) $ mkdir a/b/

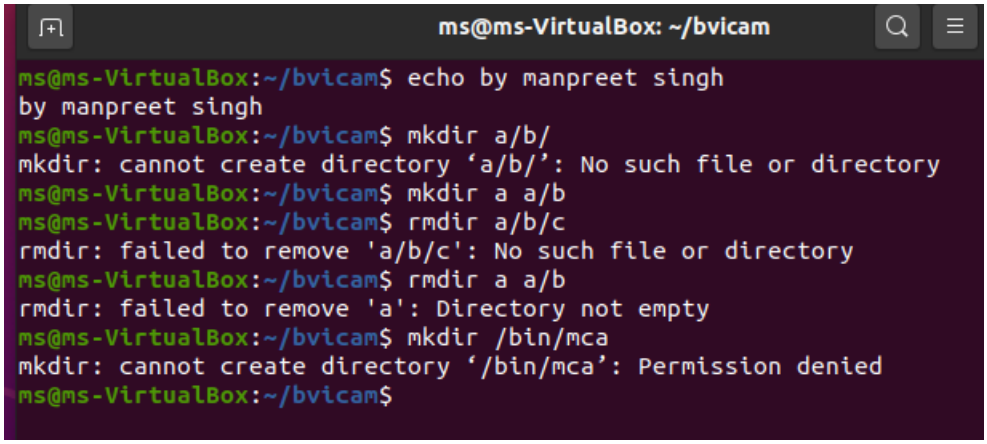
(b) $ mkdir a a/b

(c) $ rmdir a/b/c

(d) $ rmdir a a/b

(e) $ mkdir /bin/mca

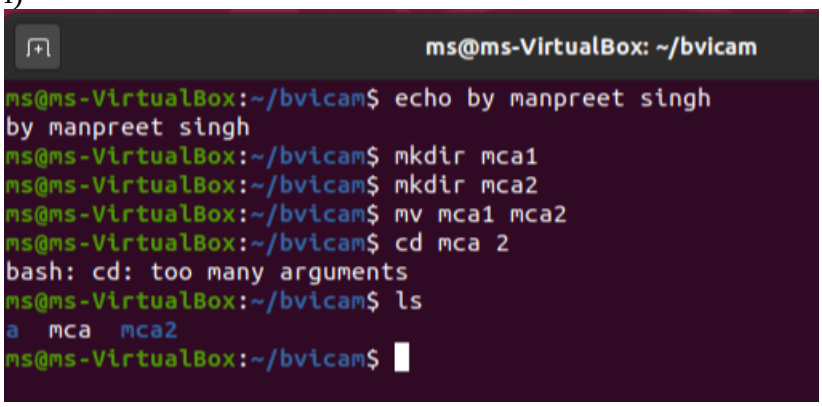
SCREENSHOT:



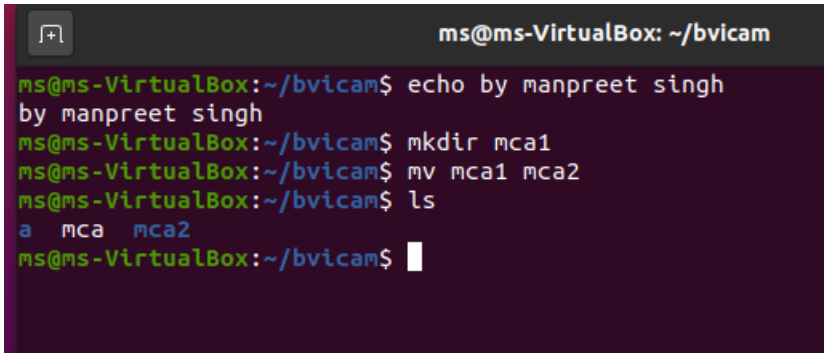
P15. How does the command mv mca1 mca2 behave, where both mca1 and mca2 are directories, when (i) mca2 exists, (ii) mca2 doesn‟t exist?

SCREENSHOT:

i)



2



P16 Assuming that you are positioned in the directory /home/bvicam, what are these commands presumed to do, and explain whether they will work at all:

(a) $ cd ../..

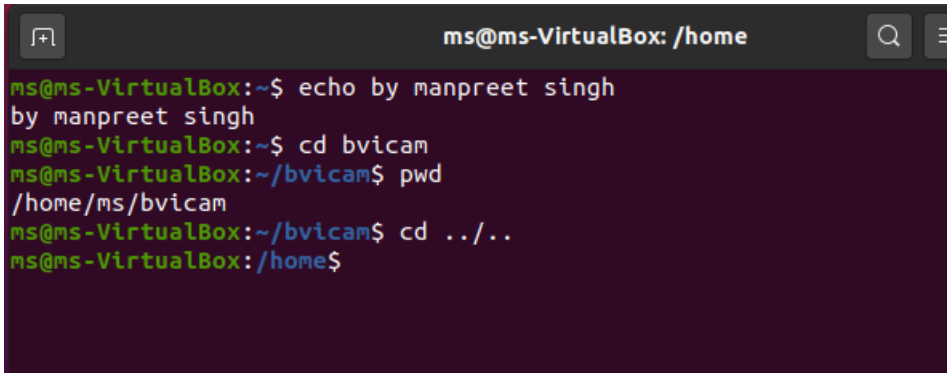
(b) $ mkdir ../bin

(c) $ rmdir ..

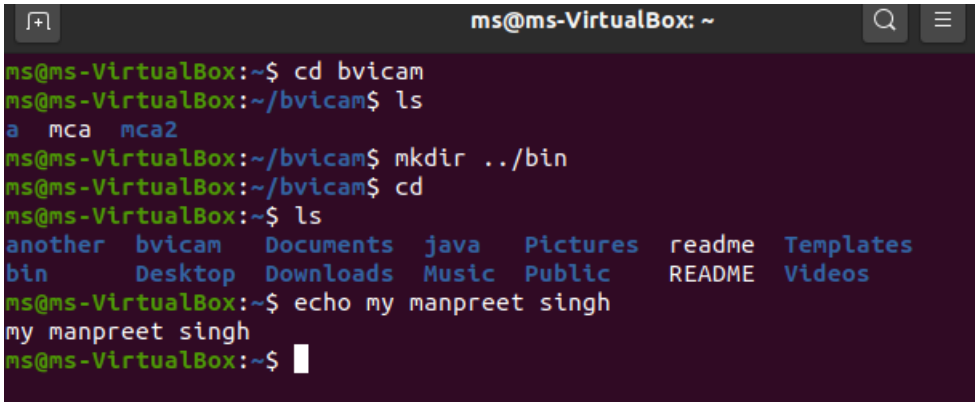
(d) $ ls ..

SCREENSHOT:

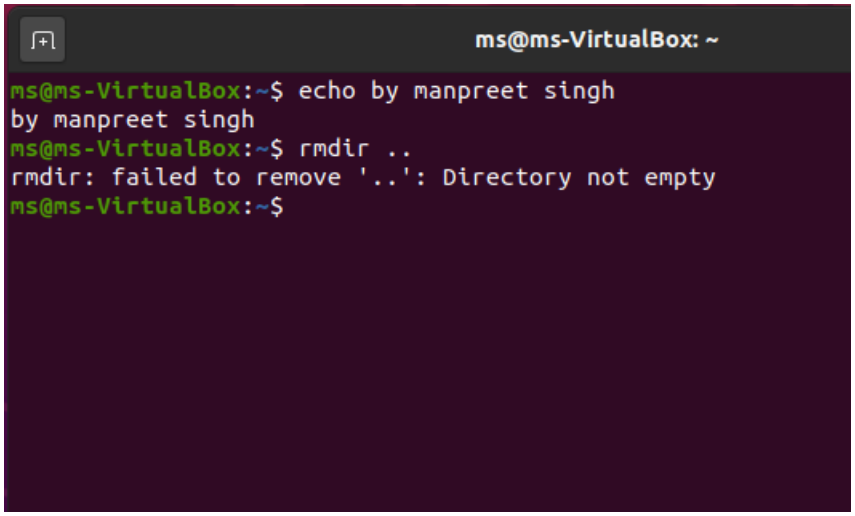
a)


ms@ms-VirtualBox: /home
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ cd bvicam
ms@ms-VirtualBox:~/bvicam$ pwd
/home/ms/bvicam
ms@ms-VirtualBox:~/bvicam$ cd ../..
ms@ms-VirtualBox:/home$

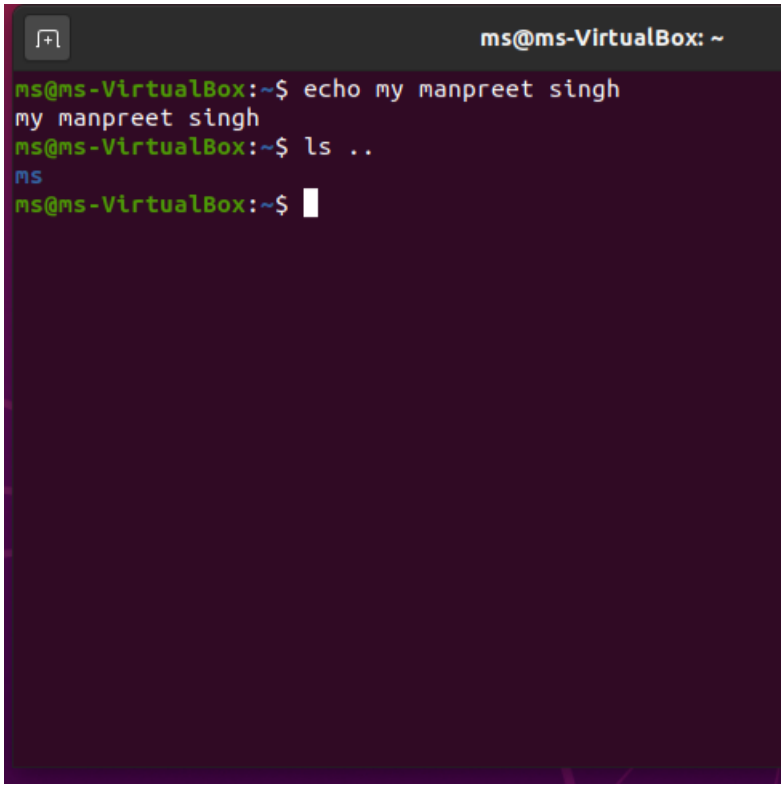
B


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ cd bvicam
ms@ms-VirtualBox:~/bvicam$ ls
a mca mca2
ms@ms-VirtualBox:~/bvicam$ mkdir ../bin
ms@ms-VirtualBox:~/bvicam$ cd
ms@ms-VirtualBox:~$ ls
another bvicam Documents java Pictures readme Templates
bin Desktop Downloads Music Public README Videos
ms@ms-VirtualBox:~$ echo my manpreet singh
my manpreet singh
ms@ms-VirtualBox:~$

C


ms@ms-VirtualBox: ~
ms@ms-VirtualBox:~$ echo by manpreet singh
by manpreet singh
ms@ms-VirtualBox:~$ rmdir ..
rmdir: failed to remove '..': Directory not empty
ms@ms-VirtualBox:~$

D



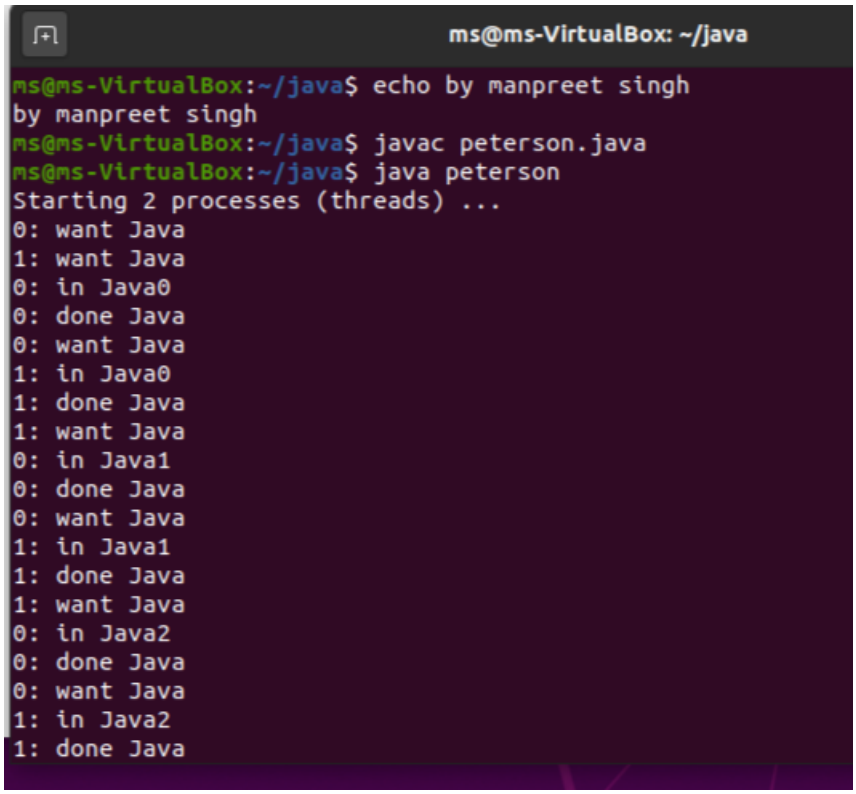
P17. Apply Peterson algorithm for solving the critical section problem with C/Java multi-threaded programming. Assume appropriate code snippet for critical section.

```
class peterson {
  static boolean[] flag = {
    false,
    false
  };
  static int turn = 0;
  static int N = 4;
  static Thread process(int i) {
    return new Thread(() -> {
      int j = 1 - i;
```

```
    for (int n = 0; n < N; n++) {
      log(i + ": want Java"); // LOCK
      flag[i] = true; // 1
      turn = j; // 2
      while (flag[j] && turn == j)
        Thread.yield(); // 3
      log(i + ": in Java" + n);
      sleep(1000 * Math.random()); // 4
      log(i + ": done Java"); // UNLOCK
      flag[i] = false; // 5
    }
  });
 }
 public static void main(String[] args) {
  try {
    log("Starting 2 processes (threads) ...");
    Thread p0 = process(0);
    Thread p1 = process(1);
    p0.start();
    p1.start();
    p0.join();
    p1.join();
  } catch (InterruptedException e) {}
 }
 static void sleep(double t) {
  try {
```

```
      Thread.sleep((long) t);
    } catch (InterruptedException e) {}
  }
  static void log(String x) {
    System.out.println(x);
  }
}
```

SCREENSHOT:



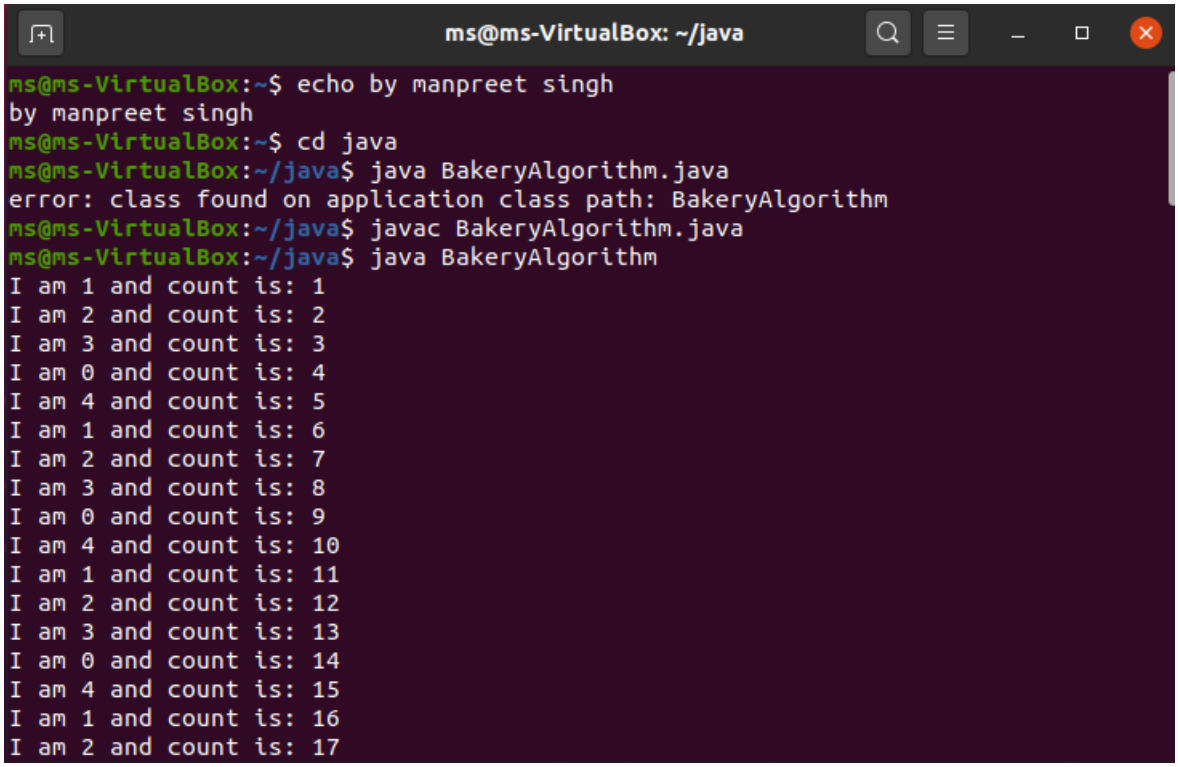
P18. Apply Bakery algorithm for synchronization of processes/threads in a C/Java program. Assume appropriate code snippet for critical section.

SOLUTION:

```
public class BakeryAlgorithm extends Thread {
  // Variables for the threads.
  public int thread_id; // The id of the current thread.
  public static final int countToThis = 200;
  public static final int numberOfThreads = 5;
  public static volatile int count = 0; // A simple counter for the testing.
  // Global variables for the bakery's algorithm.
  private static volatile boolean[] choosing = new boolean[numberOfThreads];

  private static volatile int[] ticket = new int[numberOfThreads];
  /*
   * Thread constructor.
   */
  public BakeryAlgorithm(int id) {
    thread_id = id;
  }
  // Simple test of a global counter.
  public void run() {
    int scale = 2;
    for (int i = 0; i < countToThis; i++) {
      lock(thread_id);
      // Start of critical section.
      count = count + 1;
      System.out.println("I am " + thread_id + " and count is: " +
        count);

      try {
        sleep((int)(Math.random() * scale));
      } catch (InterruptedException e) {
        /* nothing */ }
      // End of critical section.
      unlock(thread_id);
    } // for
  } // run method
  /*
   * Method that does the lock of the bakery's algorithm.
   */
  public void lock(int id) {
    choosing[id] = true;
    // Find the max value and add 1 to get the next available ticket.

    ticket[id] = findMax() + 1;
    choosing[id] = false;
    // System.out.println("Thread " + id + " got ticket in Lock");
    for (int j = 0; j < numberOfThreads; j++) {
      // If the thread j is the current thread go the next thread.
      if (j == id)
        continue;
      // Wait if thread j is choosing right now.
      while (choosing[j]) {
        /* nothing */ }
```

```
      while (ticket[j] != 0 && (ticket[id] > ticket[j] || (ticket[id] ==
         ticket[j] && id > j))) {
       /* nothing */ }
    } // for
  }
  /*
   * Method that leaves the lock.
   */
  private void unlock(int id) {
    ticket[id] = 0;
    // System.out.println("Thread " + id + " unlock");
  }
  /*
   * Method that finds the max value inside the ticket array.
   */
  private int findMax() {
    int m = ticket[0];
    for (int i = 1; i < ticket.length; i++) {
      if (ticket[i] > m)
        m = ticket[i];
    }
    return m;
  }
  public static void main(String[] args) {
    // Initialization of the global variables (it is not necessary at all).
    for (int i = 0; i < numberOfThreads; i++) {
      choosing[i] = false;
      ticket[i] = 0;
    }
    BakeryAlgorithm[] threads = new BakeryAlgorithm[numberOfThreads]; // Array of
threads.
    // Initialize the threads.
    for (int i = 0; i < threads.length; i++) {
      threads[i] = new BakeryAlgorithm(i);
      threads[i].start();
    }
    // Wait all threads to finish.
    for (int i = 0; i < threads.length; i++) {
      try {

        threads[i].join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
    System.out.println("\nCount is: " + count);
    System.out.println("\nExpected was: " + (countToThis *
      numberOfThreads));
  }
}
```

Screenshot-



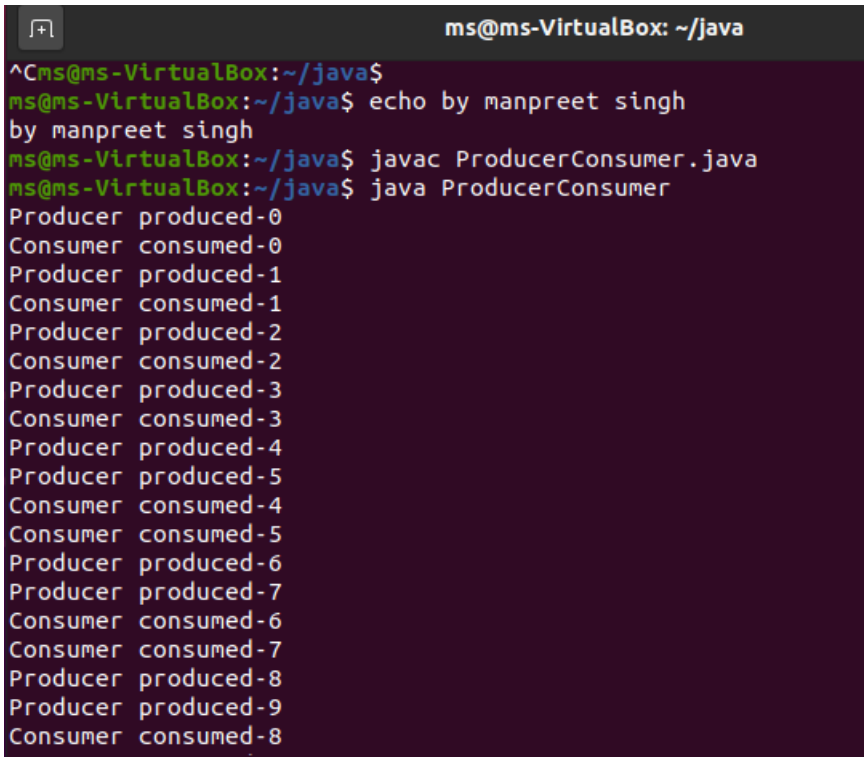
P19. Write C/Java program to simulate and solve the Producer-Consumer problem.

SOLUTION:

```
import java.util.LinkedList;
public class ProducerConsumer {
  public static void main(String[] args)
  throws InterruptedException {
    final PC pc = new PC();
    Thread t1 = new Thread(new Runnable() {
      @Override
      public void run() {
        try {
          pc.produce();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    });
    // Create consumer thread
    Thread t2 = new Thread(new Runnable() {
      @Override
      public void run() {
        try {
          pc.consume();
        } catch (InterruptedException e) {
```

```
        e.printStackTrace();
      }
    }
  });
  // Start both threads
  t1.start();
  t2.start();
  // t1 finishes before t2
  t1.join();
```

Screenshot-



P20. Implement Semaphore(s) in a C/Java-multithreaded program to simulate the working and solution of Reader-Writer problem. Assume multiple readers and writers.

SOLUTION:

```
import java.util.concurrent.*;

class Shared {

  static int count = 0;
```

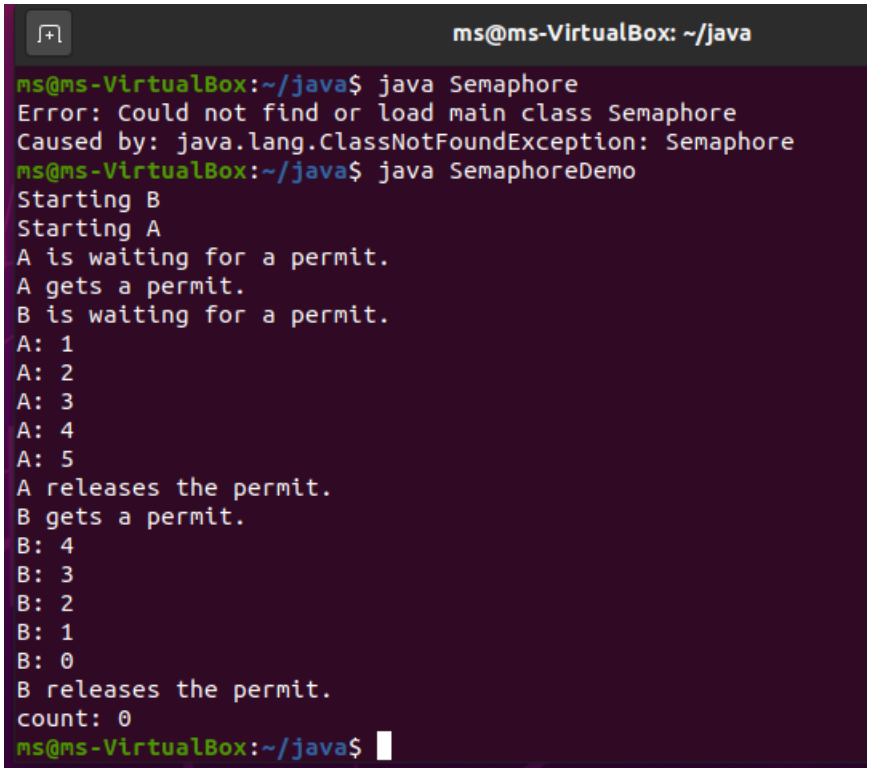
```
}
class MyThread extends Thread {
  Semaphore sem;
  String threadName;
  public MyThread(Semaphore sem, String threadName) {
    super(threadName);
    this.sem = sem;
    this.threadName = threadName;
  }
  @Override
  public void run() {
    // run by thread A
    if (this.getName().equals("A")) {
      System.out.println("Starting " + threadName);
      try {
        // First, get a permit.
        System.out.println(threadName + " is waiting for a permit.");
        // acquiring the lock
        sem.acquire();
        System.out.println(threadName + " gets a permit.");
        for (int i = 0; i < 5; i++) {
          Shared.count++;
          System.out.println(threadName + ": " + Shared.count);
          // Now, allowing a context switch -- if possible.
          // for thread B to execute
          Thread.sleep(10);
```

```
        }
      } catch (InterruptedException exc) {
        System.out.println(exc);
      }
      // Release the permit.
      System.out.println(threadName + " releases the permit.");
      sem.release();

    }
    // run by thread B
    else {
      System.out.println("Starting " + threadName);
      try {
        // First, get a permit.
        System.out.println(threadName + " is waiting for a permit.");
        // acquiring the lock
        sem.acquire();
        System.out.println(threadName + " gets a permit.");
        // Now, accessing the shared resource.
        // other waiting threads will wait, until this
        // thread release the lock
        for (int i = 0; i < 5; i++) {
          Shared.count--;
          System.out.println(threadName + ": " + Shared.count);
          // Now, allowing a context switch -- if possible.
          // for thread A to execute
```

```
          Thread.sleep(10);
        }
      } catch (InterruptedException exc) {
        System.out.println(exc);
      }
      // Release the permit.
      System.out.println(threadName + " releases the permit.");
      sem.release();
    }
  }
}
// Driver class
public class SemaphoreDemo {
  public static void main(String args[]) throws InterruptedException {
    // creating a Semaphore object
    // with number of permits 1
    Semaphore sem = new Semaphore(1);
    // creating two threads with name A and B
    // Note that thread A will increment the count
    // and thread B will decrement the count
    MyThread t1 = new MyThread(sem, "A");
    MyThread t2 = new MyThread(sem, "B");
    // stating threads A and B
    t1.start();
    t2.start();
    // waiting for threads A and B
```

```
    t1.join();


    t2.join();
    System.out.println("count: " + Shared.count);
  }
}
```

Screenshot-



P21 Create a zombie process and an orphan process in a C program with appropriate system calls.

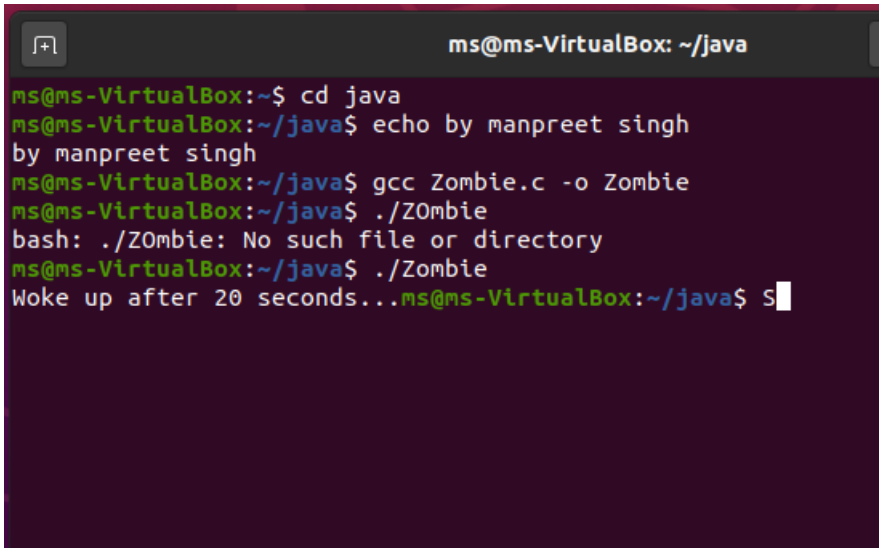
SOLUTION:

Zombie Process:

```
#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>
```

```
#include<stdio.h>

int main() {
  // Fork returns process id
  // in parent process
  pid_t child_pid = fork();
  // Parent process
  if (child_pid > 0)
    sleep(20);
  // Child process
  else
    exit(0);
  printf("Woke up after 20 seconds...");
  return 0;
}
```

SCREENSHOT:

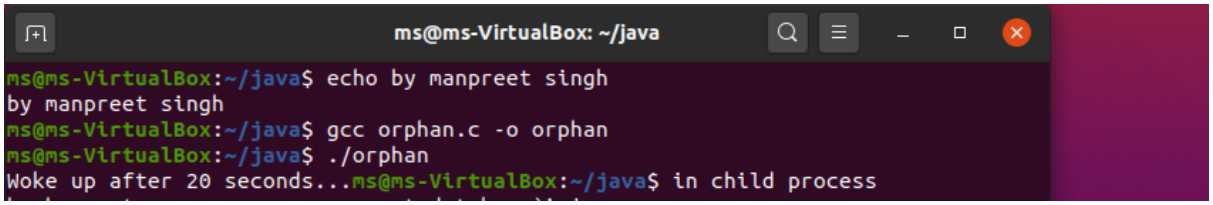


Orphan Process

```
#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include<stdio.h>

int main()

{
```

```
// Fork returns process id

// in parent process

pid_t child_pid = fork();

// Parent process

if (child_pid > 0)

sleep(20);

// Child process

else

exit(0);

printf("Woke up after 20 seconds...");

return 0;

}
```

SCREENSHOT:



P22. Write a „C“ program which creates a new process and allows both, child and parent, to report their identification numbers (ids). The parent process should wait for the termination of the child process.

SOLUTION:

```
#include<stdio.h>

#include<fcntl.h>
```

```
#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

int main() {
   pid_t pid;
   pid = fork();
   if (pid == 0) {
     printf("\nI'mInChild..");
     printf("\nChild Process Id:%d", getpid());
     printf("\nChildClosed");
     exit(0);
   } else {
     wait(NULL);
     printf("\nFrom Parent Process..");
     printf("\nParent Process Id: %d", getpid());
     printf("\nParent's Child Id: %d", pid);
     printf("\nParentClosed");
   }
   printf("\n");
   return 0;
```

SCREENSHOT:

P23. Write two „C“ programs (A.c and B.c) where one program (A.c) creates a child process and then that child process executes the code of

other program (B.c). The logic of program „B.c" is to generate all the prime numbers within the specified limit.

SOLUTION:

A.c

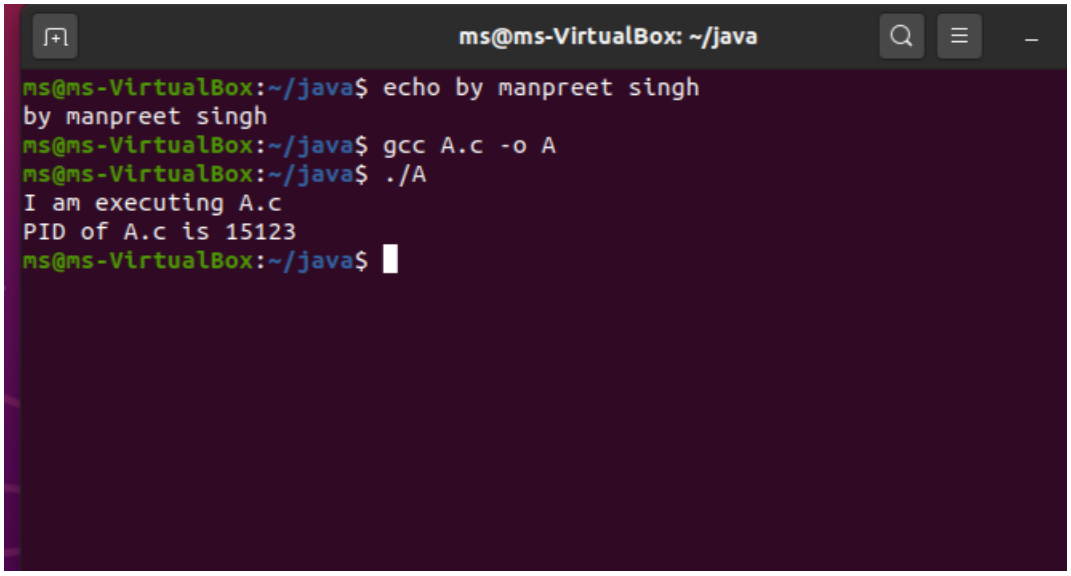
```
#include<stdio.h>
#include<unistd.h>
int main(){
printf("I am executing A.c \n");
printf("PID of A.c is %d\n",getpid());
char *args[]={"./B.c",NULL};
execv(args[0],args);
}
```

B.c

```
 #include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>
void main()
{
int n;
printf("\nUp to How Many Numbers:");
scanf("%d",&n);

for(int i=1; i<n; i++)
```

```
{
int flag=0;
for(int j=2; j<=i/2; j++)
{
if(i%j==0)
{
flag=1;
break;
}
}
if(i==1)
{
printf("\n1 is neither Prime nor Composite..\n");
}
else
{
if(flag==0)
{
printf("%d is a Prime Number..\n",i);
}
}
}
}
```

SCrenshot-



P24. Write an appropriate „C" program which implements the concept of dynamic memory allocation (use of malloc(), calloc(), realloc(), and free() system call).

SOLUTION:

Malloc:

#include<stdio.h> #include<malloc.h> #include<stdlib.h> void main()

{

int n, *ptr, i;

printf("Input array size: "); scanf("%d",&n);

ptr = (int *)malloc(n*sizeof(int)); if(ptr==NULL)

{

printf("\nNo Allocation of memory");

}

else

{

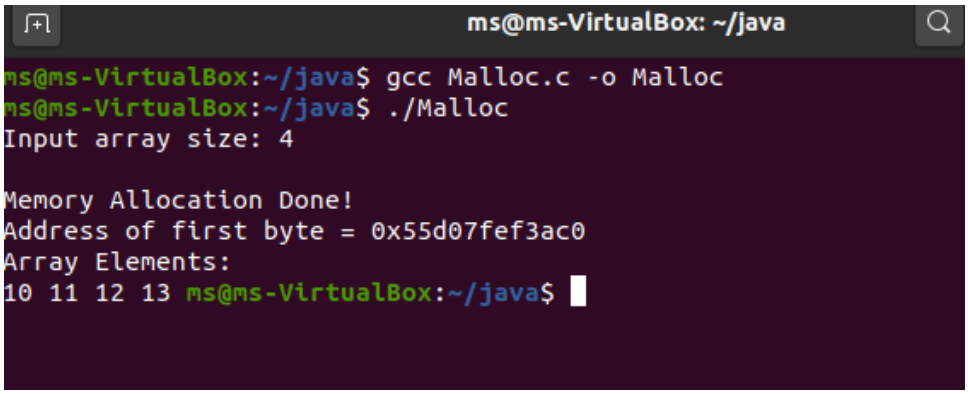
printf("\nMemory Allocation Done!"); printf("\nAddress of first byte = %p", ptr); for(i=0;

i<n; i++)

{

```
ptr[i] = i+10;

}

}

printf("\nArray Elements: \n"); for(i=0; i<n; i++)

{

printf("%d ", ptr[i]);

// printf("%p ", ptr+i);

}
```

SCREENSHOT:

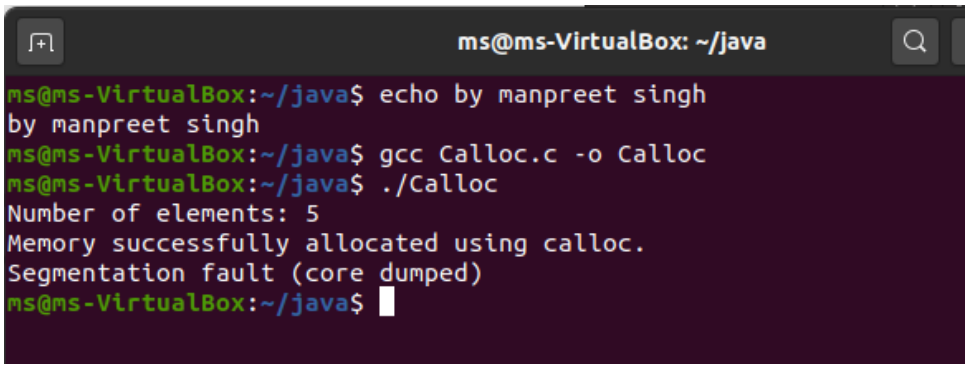


Calloc:

```
#include <stdio.h> #include <stdlib.h> int main()

{

int* ptr; int n, i; n = 5;

printf("Number of elements: %d\n", n);

// Dynamically allocate memory using calloc() ptr = (int*)calloc(n, sizeof(int));

// Check if the memory has been successfully

// allocated by calloc or not if (ptr == NULL) {

printf("Memory not allocated.\n"); exit(0);

}
```

```
else {
printf("Memory successfully allocated using calloc.\n");
// Get the elements of the array for (i = 0; i < n; ++i) {
ptr[i] = i + 1;
}
// Print the elements of the array printf("The elements of the array are: "); for (i = 0; i < n;
++i) {
printf("%d, ", ptr[i]);
}
}
return 0;
}
```

SCREENSHOT:



Realloc:

#include <stdio.h> #include <stdlib.h> int main()

{

int* ptr; int n, i;

// Get the number of elements for the array n = 5;

printf("Number of elements: %d\n", n);

```
// Dynamically allocate memory using calloc() ptr = (int*)calloc(n, sizeof(int));
if (ptr == NULL) { printf("Memory not allocated.\n"); exit(0);
}
else {
// Memory has been successfully allocated printf("Memory successfully allocated using
calloc.\n");
// Get the elements of the array for (i = 0; i < n; ++i) {
ptr[i] = i + 1;
}

// Print the elements of the array printf("The elements of the array are: "); for (i = 0; i < n;
++i) {
printf("%d, ", ptr[i]);
}
n = 10;
printf("\n\nThe new size of the array: %d\n", n);
// Dynamically re-allocate memory using realloc() ptr = realloc(ptr, n * sizeof(int));
// Memory has been successfully allocated
printf("Memory successfully re-allocated using realloc.\n");
// Get the new elements of the array for (i = 5; i < n; ++i) {
ptr[i] = i + 1;
}
// Print the elements of the array printf("The elements of the array are: "); for (i = 0; i < n;
++i) {
printf("%d, ", ptr[i]);
}
```

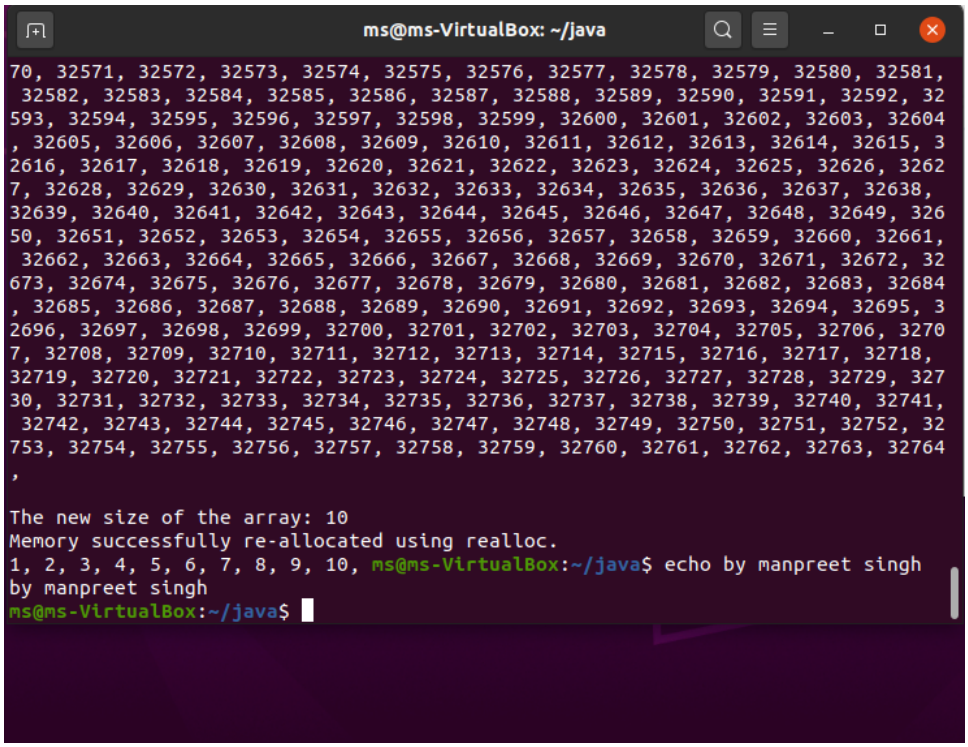
free(ptr);

}

return 0;

}

Screenshot-



Free:

#include <stdio.h> #include <stdlib.h> int main()

{

// This pointer will hold the

// base address of the block created int *ptr, *ptr1;

int n, i; n = 5;

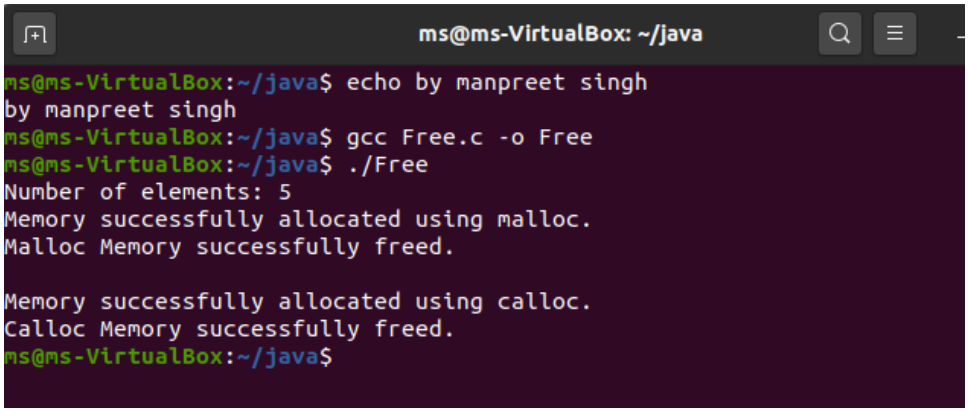
printf("Number of elements: %d\n", n);

// Dynamically allocate memory using malloc() ptr = (int*)malloc(n * sizeof(int));

```
// Dynamically allocate memory using calloc() ptr1 = (int*)calloc(n, sizeof(int));


// Check if the memory has been successfully
// allocated by malloc or not
if (ptr == NULL || ptr1 == NULL) { printf("Memory not allocated.\n"); exit(0);
}
else {
// Memory has been successfully allocated printf("Memory successfully allocated using
malloc.\n");
// Free the memory free(ptr);
printf("Malloc Memory successfully freed.\n");
// Memory has been successfully allocated printf("\nMemory successfully allocated using
calloc.\n");
// Free the memory free(ptr1);
printf("Calloc Memory successfully freed.\n");
}


return 0;
}#include <stdio.h> #include <stdlib.h> int main()
{
// This pointer will hold the
// base address of the block created int *ptr, *ptr1;
int n, i; n = 5;


printf("Number of elements: %d\n", n);
// Dynamically allocate memory using malloc() ptr = (int*)malloc(n * sizeof(int));
```

```
// Dynamically allocate memory using calloc() ptr1 = (int*)calloc(n, sizeof(int));
// Check if the memory has been successfully
// allocated by malloc or not
if (ptr == NULL || ptr1 == NULL) { printf("Memory not allocated.\n"); exit(0);
}
else {
// Memory has been successfully allocated printf("Memory successfully allocated using
malloc.\n");
// Free the memory free(ptr);
printf("Malloc Memory successfully freed.\n");
// Memory has been successfully allocated printf("\nMemory successfully allocated using
calloc.\n");
// Free the memory
free(ptr1);
printf("Calloc Memory successfully freed.\n");
}
return 0;
}
```

SCREENSHOT:



P25. Create a text file, named as „courses.txt“ that contains the following four lines:

Java Programming Operating System Discrete Structure

Write a „C“ program that forks three other processes. After forking, the parent process goes into wait state and waits for the children to finish their execution. Each child process reads a line from the „course.txt“ file (Child 1 Reads Line 1, Child 2 Reads Line 2, and Child 3 Reads Line 3) and each prints the respective line. The lines can be printed in any order.

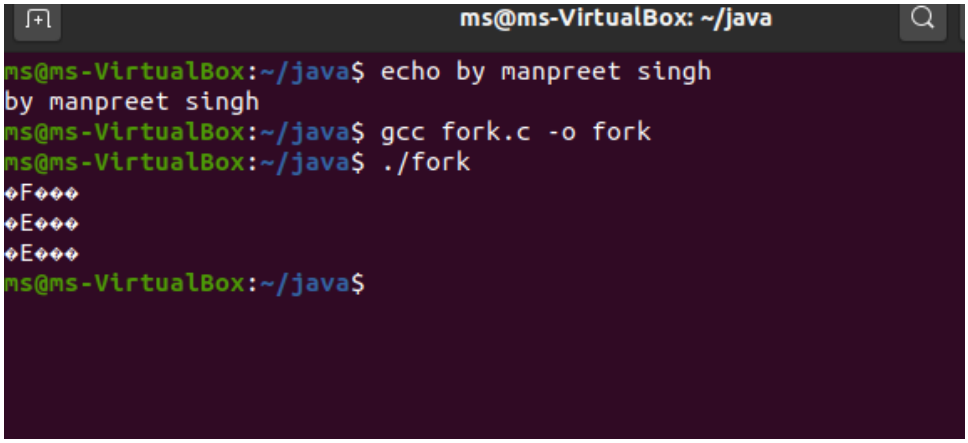
SOLUTION:

```
#include<fcntl.h> #include<unistd.h> #include<sys/stat.h> #include<stdlib.h>

#include<sys/wait.h> #include<stdio.h> void main()

{

pid_t pid; int fd;

int linecount[3]={16,17,21}; int startpoint[3]={0,17,33}; for(int i=0;i<3;i++)

{

pid=fork(); if(pid==0)

{

char read_value[linecount[i]]; fd=open("Courses.txt",O_RDWR);

lseek(fd,startpoint[i],SEEK_SET); read(fd, read_value,linecount[i]);

printf("%s\n",read_value); close(fd);

exit(0);

}

else

{

wait(NULL);
```

}

}

SCREENSHOT:



P26. Write a „C“ program (using appropriate system calls of Linux) that generates „n“ integers and stores them in a text file, named as „All.txt“. Then, retrieve the stored integers from this file and copy to “Odd.txt” and

„Even.txt“ based upon the type of number, i.e. if the retrieved integer if odd number then store in „Odd.txt“ file or if the retrieved integer is even then store in „Even.txt“ file. Finally, display the contents of all three files on the screen.

SOLUTION:

#include<stdio.h> #include<fcntl.h> #include<unistd.h> #include<sys/stat.h> void main()

int n;

printf("\nUp to How Many Numbers: "); scanf("%d",&n);

char write_value[n], read_value[n]; int fd,fd_odd,fd_even; printf("\nWriting To All.txt: ");

for(int i=0;i<n;i++)

{

write_value[i]=(i+1)+'0'; printf("%c",write_value[i]);

```
}
printf("\n");
fd=open("All.txt",O_CREAT | O_WRONLY, 0777); write(fd,write_value,n);
close(fd);
fd=open("All.txt", O_RDWR); read(fd,read_value,n);
char write_odd[n/2],write_even[n/2]; int ind_odd=0,ind_even=0;
for(int i=0;i<n;i++)
{
int temp=read_value[i]; if(temp%2==0)

write_even[ind_even++]=read_value[i];
}
else
{
write_odd[ind_odd++]=read_value[i];
}
}
printf("\nWriting To odd.txt:"); for(int j=0;j<n/2;j++)
{
printf("%c",write_odd[j]);
}
printf("\n");
printf("\nWriting To even.txt:"); for(int j=0;j<n/2;j++)
{
printf("%c",write_even[j]);
}
```

printf("\n");

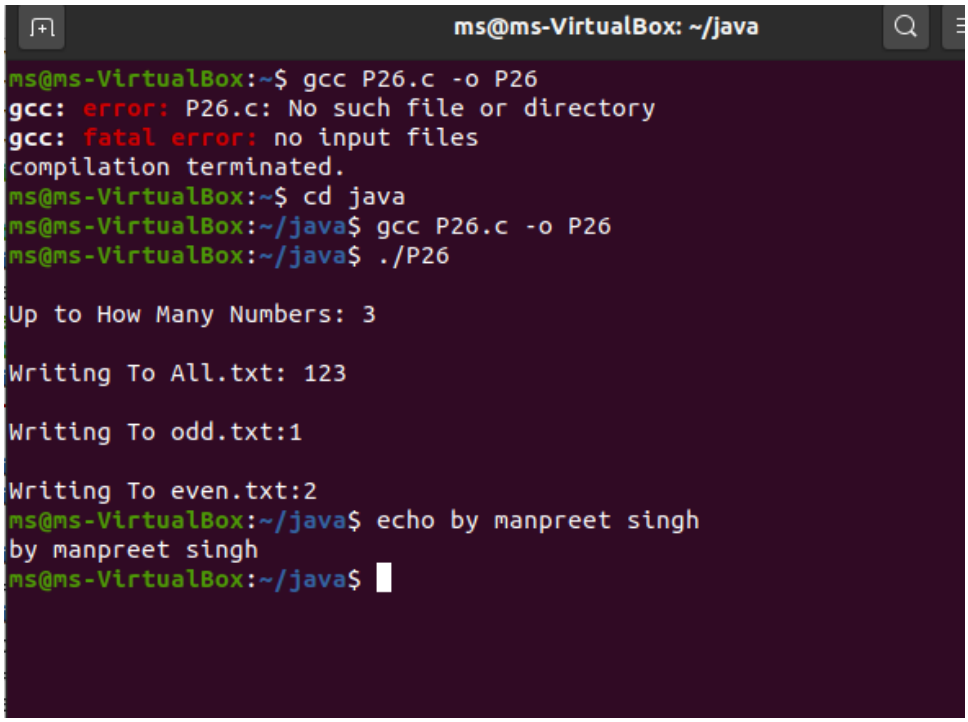
fd_odd=open("odd.txt",O_CREAT | O_RDWR, 0777); write(fd_odd, write_odd, n/2);

close(fd_odd); fd_even=open("even.txt",O_CREAT | O_RDWR, 0777); write(fd_even,

write_even, n/2); close(fd_even);

}

SCREENSHOT:



P27. Write a program in „C“ which accepts the file or directory name and permission (access rights) from the user and then changes the access rights accordingly. Use appropriate system call(s) of Linux.

SOLUTION:

#include<stdio.h> #include<fcntl.h> #include<unistd.h> #include<sys/stat.h> int main(){

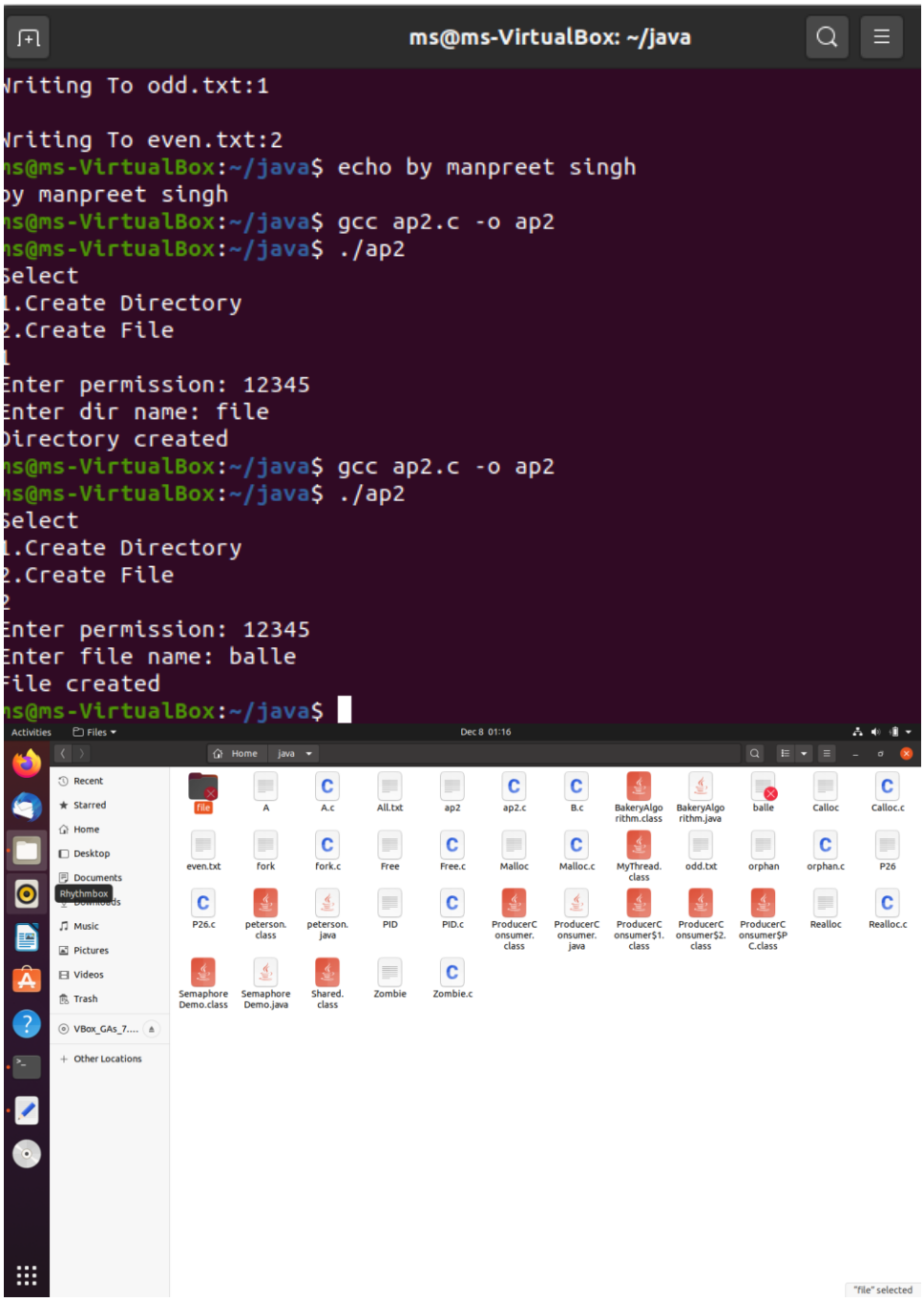
int inp,check,fd; unsigned int perm; char name[51] = ""; printf("Select\n1.Create

Directory\n2.Create File\n"); scanf("%d", &inp);

printf("Enter permission: "); scanf("%o", &perm); switch(inp)

```
{
case 1: printf("Enter dir name: "); scanf("%50s", name);
check = mkdir(name, perm); if (!check)
{
printf("Directory created\n");
}
else
{
printf("Unable to create directory\n");
}
break;
case 2: printf("Enter file name: "); scanf("%50s", name);
fd = open(name,O_CREAT | O_RDWR, perm); printf("File created\n");
close(fd);
break;
}
}
```

Screenshot

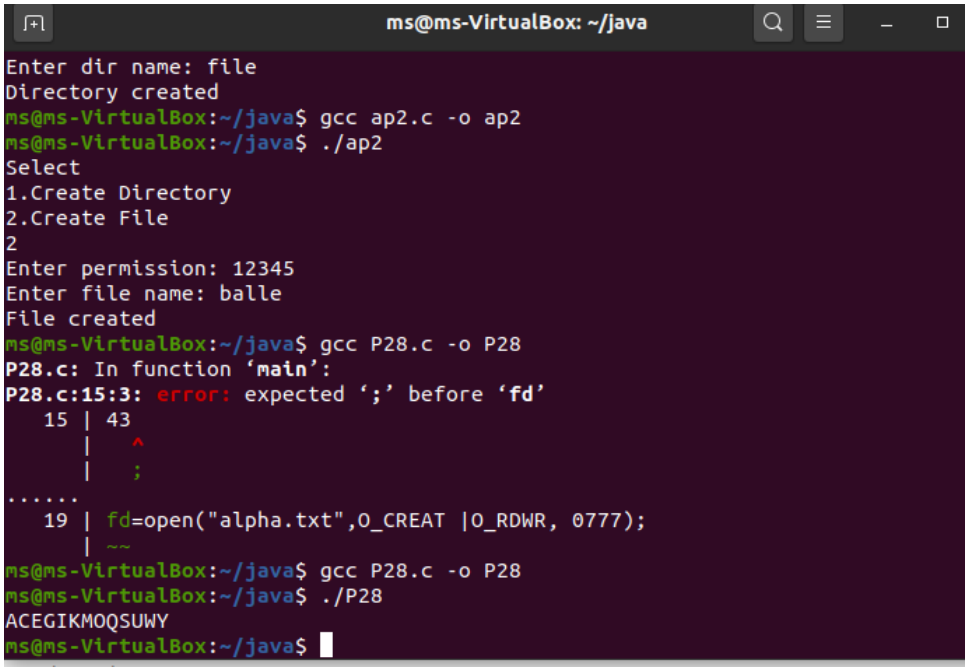


P28. Write a „C“ program (using appropriate system calls of Linux) which generates and stores the characters from „a“ to „z“. Then, display the stored characters in alternative manner, like: a, c, e, g, ..., etc.

SOLUTION:

```
#include<stdio.h> #include<fcntl.h> #include<unistd.h>

#include<sys/stat.h> void main()
{
int n=26,ind=0;
int fd,sizeRead,sizeWrite;
char write_value[n], read_value[n]; for(int i=65;i<91;i++)
{
write_value[ind]=i; ind++;
}

fd=open("alpha.txt",O_CREAT |O_RDWR, 0777); sizeWrite=write(fd, write_value,n);
close(fd); fd=open("alpha.txt",O_RDWR); sizeRead=read(fd,read_value,n); close(fd);
for(int i=0;i<n;i+=2)
{
printf("%c",read_value[i]);
}
printf("\n");
}
```

SCREENSHOT:



P29.Write a „C" program (using appropriate system calls of Linux) which receives roll number and names of „n" students, from the user one-by-one and then stores them in a text file, named as „Student.txt". After inserting all „n" roll numbers and names, display the contents of file. Also, display the access rights of the file „Student.txt".

SOLUTION:

#include <stdio.h> #include <stdlib.h> struct student {

int rollno;

char name[200];

};

int writef(FILE *file, struct student *s) { int out = fprintf(

file, "%d\n%s\n",

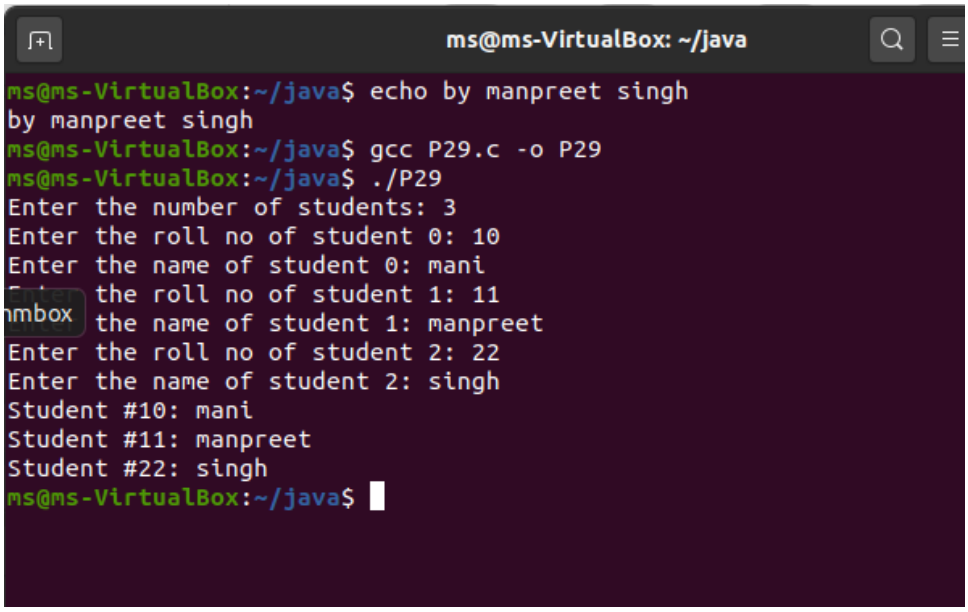
s->rollno, s->name

);

fflush(file); return out;

```
}
int readf(FILE *file, struct student *s) {
if (fscanf(file, "%d", &s->rollno) == EOF) return 0; if (fscanf(file, "%s", s->name) == EOF)
return 0; return 1;
}
int main() { int n;
printf("Enter the number of students: "); scanf("%d", &n);
struct student s;
FILE *file = fopen("Student.txt", "w"); for (int i = 0; i < n; i++) {
printf("Enter the roll no of student %d: ", i); scanf("%d", &s.rollno);
printf("Enter the name of student %d: ", i); scanf("%s", s.name);
writef(file, &s);
}
fclose(file);
file = fopen("Student.txt", "r"); int i = 0;
while (readf(file, &s)) {
printf("Student #%d: %s\n", s.rollno, s.name); if(i == 5) break;
i++;
}
fclose(file); return 0;
}
```

SCREENSHOT:



P30. Demonstrate the use of following system calls by writing an appropriate „C“ program.

1. lseek()

(b) chmod()

2. umask()

(d) access()

3. utime()

a)

SOLUTION

#include<unistd.h> #include<stdio.h> #include<fcntl.h> int main()

{

int fd;

char buffer[80];

char msg[50]="Hello i am Anjali."; fd=open("check.txt",O_RDWR); printf("fd=%d",fd);

if(fd!=-1)

{

printf("\ncheck.txt opened wih read write access\n"); write(fd,msg,sizeof(msg));

lseek(fd,0,SEEK_SET); read(fd,buffer,sizeof(msg));

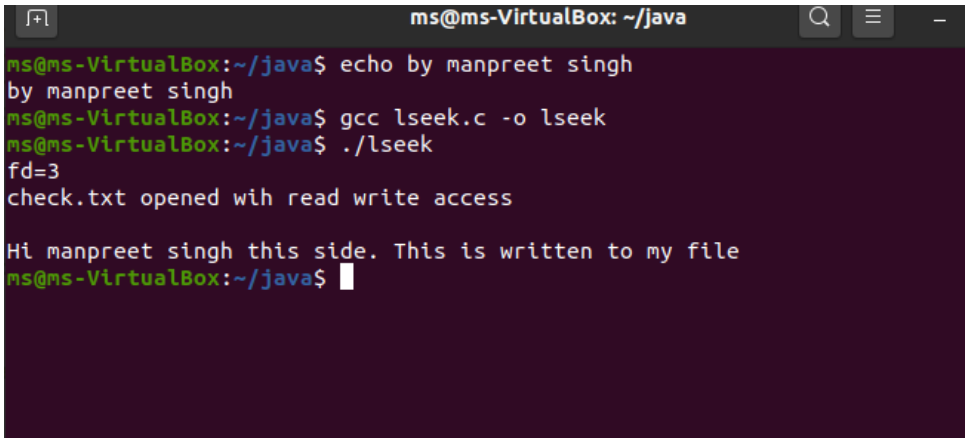
printf("\n%s This is written to my file\n",buffer); close(fd);

}

return 0;

}

SCREENSHOT:



Lseek() (C System Call): lseek is a system call that is used to change the location of the read/write pointer of a file descriptor. The location can be set either in absolute or relative terms.

b. SOLUTION:

#include <stdio.h> #include <stdlib.h> #include <string.h> #include <errno.h> #include

<sys/stat.h>

int main(int argc,char **argv)

{

char mode[]="0777";

char buf[100]="/home/anjali/First.c"; int i;

i=strtol(mode,0,8); if(chmod(buf,i)<0)

```
{

fprintf(stderr,"%s:error in chmod(%s,%s)-%d(%s)\n",

argv[0],buf,mode,errno,strerror(errno));

exit(1);

}

return(0);

}
```

SCREENSHOT:

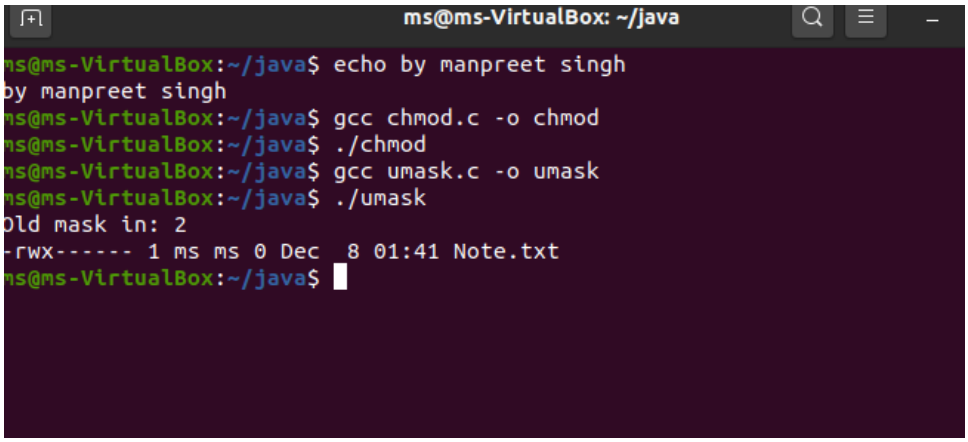
chmod(): In Unix-like operating systems, the chmod command is used to change the access mode of a file. The name is an abbreviation of change mode. Syntax : chmod [reference][operator][mode] file... The references are used to distinguish the users to whom the permissions apply i.e. they are list of letters that specifies whom to give permissions. The references are represented by one or more of the following letters: Reference Class Description u owner file's owner g group users who are members of the file's group.

C
SOLUTION:

#define _POSIX_SOURCE #include <fcntl.h>

#include <sys/stat.h> #include <unistd.h> #include <stdlib.h>

```
#undef _POSIX_SOURCE

#include <stdio.h>

int main(){ int fd; mode_t old;

printf("Old mask in: %i\n", old=umask(S_IRWXG)); if((fd = creat("Note.txt",

S_IRWXU|S_IRWXG)) < 0){ perror("creat() error");

}

else{

system("ls -l Note.txt"); close(fd); unlink("Note.txt");

}

umask(old);

}
```



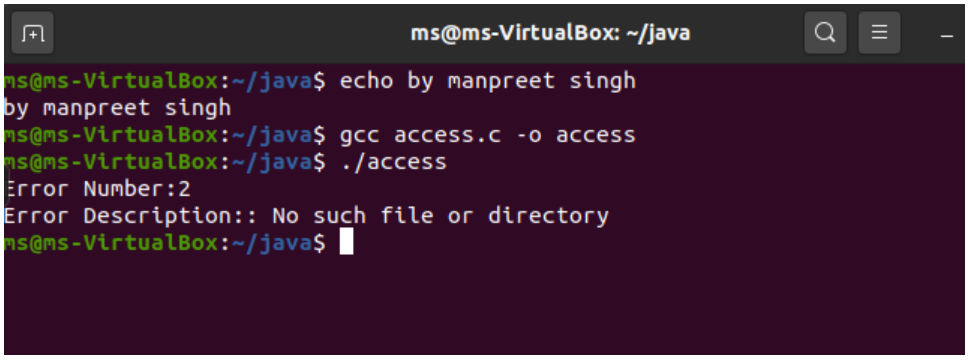
umask() : On Linux and other Unix-like operating systems, new files are created with a default set of permissions. Specifically, a new file's permissions may be restricted in a specific way by applying a permissions "mask" called the umask. The umask command is used to set this mask, or to show you its current value.

b. SOLUTION:

```
#include<stdio.h> #include<unistd.h> #include<errno.h> #include<sys/types.h>

#include<sys/stat.h> #include<fcntl.h> extern int errno;


int main(int argc,const char *argv[])
```

```
{
int fd=access("Courses.txt",F_OK); if(fd==-1)
{
printf("Error Number:%d\n",errno); perror("Error Description:");
}
else
printf("No error\n"); return 0;
}
```

SCREENSHOT:



access() : In Linux, the access command is used to check whether the calling program has access to a specified file. It can be used to check whether a file exists or not. The check is done using the calling process's real UID and GID.

b. SOLUTION:

```
#include <stdio.h> #include <string.h> #include<stdlib.h>

int main(void)
{
FILE* uptimefile; char uptime_chr[28]; long uptime=0;
if((uptimefile=fopen("/proc/uptime","r"))==NULL) perror("supt"),exit(EXIT_FAILURE);
fgets(uptime_chr, 12, uptimefile); fclose(uptimefile); uptime=strtol(uptime_chr,NULL,10);
```

```
printf("System up for %ld seconds,%ld hours \n",uptime,uptime/ 3600);

exit(EXIT_SUCCESS);

}
```

SCREENSHOT: